Higher-Order Spectral Analysis Toolbox

For Use with MATLAB®

Ananthram Swami
Jerry M. Mendel
Chrysostomos L. (Max) Nikias

Computation

Visualization

Programming

User's Guide

Version 2

How to Contact The MathWorks:



508-647-7000 Phone



508-647-7001 Fax



The MathWorks, Inc.
24 Prime Park Way
Natick, MA 01760-1500 Mail



`http://www.mathworks.com` Web
`ftp.mathworks.com` Anonymous FTP server
`comp.soft-sys.matlab` Newsgroup



`support@mathworks.com` Technical support
`suggest@mathworks.com` Product enhancement suggestions
`bugs@mathworks.com` Bug reports
`doc@mathworks.com` Documentation error reports
`subscribe@mathworks.com` Subscribing user registration
`service@mathworks.com` Order status, license renewals, passcodes
`info@mathworks.com` Sales, pricing, and general information

Higher-Order Spectral Analysis Toolbox User's Guide

© COPYRIGHT 1984 - 1998 by The MathWorks, Inc. All Rights Reserved.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

U.S. GOVERNMENT: If Licensee is acquiring the Programs on behalf of any unit or agency of the U.S. Government, the following shall apply: (a) For units of the Department of Defense: the Government shall have only the rights specified in the license under which the commercial computer software or commercial software documentation was obtained, as set forth in subparagraph (a) of the Rights in Commercial Computer Software or Commercial Software Documentation Clause at DFARS 227.7202-3, therefore the rights set forth herein shall apply; and (b) For any other unit or agency: NOTICE: Notwithstanding any other lease or license agreement that may pertain to, or accompany the delivery of, the computer software and accompanying documentation, the rights of the Government regarding its use, reproduction, and disclosure are as set forth in Clause 52.227-19 (c)(2) of the FAR.

MATLAB, Simulink, Handle Graphics, and Real-Time Workshop are registered trademarks and Stateflow and Target Language Compiler are trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Printing History: May 1993 First printing
September 1995 Second printing
January 1998 Third printing

Contents

About the Authors

1 Tutorial

Introduction **1-2**

Polyspectra and Linear Processes **1-4**
Introduction **1-4**
Definitions **1-6**
Why Do We Need Higher-Order Statistics? **1-10**
Bias and Variance of an Estimator **1-11**
Estimating Cumulants **1-12**
Examples **1-14**
Estimating Polyspectra and Cross-polyspectra **1-15**
Estimating the Power Spectrum **1-15**
Estimating Bispectra and Cross-Bispectra **1-16**
Examples **1-18**
Examples **1-19**
Estimating Bicoherence **1-20**
Examples **1-20**
Testing for Linearity and Gaussianity **1-22**
Examples **1-24**

Parametric Estimators, ARMA Models **1-26**
MA Models **1-29**
Examples **1-30**
AR Models **1-31**
Examples **1-32**
ARMA Models **1-32**
Examples **1-34**
AR Order Determination **1-34**
Examples **1-35**
MA Order Determination **1-36**
Examples **1-37**
Linear Processes: Impulse Response Estimation **1-37**
The Polycepstral Methods **1-38**
Examples **1-39**
Examples **1-41**
The Matsuoka-Ulrych Algorithm **1-41**
Examples **1-42**
Linear Processes: Theoretical Cumulants and Polyspectra .. **1-43**
Examples **1-43**
Summary **1-45**

Linear Prediction Models **1-47**
Levinson Recursion **1-47**
Trench Recursion **1-49**
Examples **1-50**
Deterministic Formulation of FBLS **1-53**
Adaptive Linear Prediction **1-54**
RIV Algorithm: Transversal Form **1-56**
Examples **1-57**
RIV Algorithm: Double-Lattice Form **1-58**
Examples **1-60**
Summary **1-61**

Harmonic Processes and DOA **1-62**
Resolution and Variance **1-65**
AR and ARMA Models **1-66**
Pisarenko's Method **1-67**
Multiple Signal Classification (MUSIC) **1-68**
Minimum-Norm Method **1-69**
ESPRIT **1-70**

Criterion-Based Estimators 1-72
Cumulant-Based Estimators 1-74
Examples 1-75
Examples 1-77
Summary 1-79

Nonlinear Processes 1-80
Solution Using Cross-Bispectra 1-80
Examples 1-82
Solution Using FTs 1-82
Examples 1-83
Quadratic Phase Coupling 1-84
Examples 1-87
Summary 1-88

Time-Frequency Distributions 1-89
Wigner Spectrum 1-90
Examples 1-93
Examples 1-94
Wigner Bispectrum 1-94
Examples 1-96
Examples 1-97
Wigner Trispectrum 1-98
Examples 1-99
Examples 1-100
Summary 1-100

Time-Delay Estimation 1-101
A Cross-Correlation Based Method 1-101
Examples 1-103
A Cross-Cumulant Based Method 1-103
Examples 1-105
A Hologram Based Method 1-105
Examples 1-107
Summary 1-107

Case Studies **1-108**
Sunspot Data **1-108**
Canadian Lynx Data **1-114**
Examples **1-114**
A Classification Example **1-120**
Laughter Data **1-122**
Pitfalls and Tricks of the Trade **1-131**

Data Files **1-134**

References **1-139**

2 Reference

Function Tables **2-2**
Higher-Order Spectrum Estimation: Conventional Methods . . **2-2**
Higher-Order Spectrum Estimation: Parametric Methods **2-3**
Quadratic Phase Coupling (QPC) **2-3**
Second-Order Volterra Systems **2-4**
Harmonic Retrieval **2-4**
Time-Delay Estimation (TDE) **2-4**
Array Processing: Direction of Arrival (DOA) **2-4**
Adaptive Linear Prediction **2-5**
Impulse Response (IR), Magnitude and
Phase Retrieval **2-5**
Time-Frequency Estimates **2-5**
Utilities **2-6**
Demo **2-6**

Miscellaneous **2-7**
Prompting **2-7**
Guided tour **2-7**
Addenda **2-7**

About the Authors

About the Authors

Ananthram Swami

Ananthram Swami received his B.Tech, M.S. and Ph.D. degrees in electrical engineering from the Indian Institute of Technology at Bombay, Rice University, and the University of Southern California, respectively. He has held positions with Unocal, USC, CS-3 and Malgudi Systems. He is currently a Senior SCEEE Research Fellow at the Army Research Lab, Adelphi, MD.

Dr. Swami has published over fifty journal and conference papers in the areas of modeling and parameter estimation of non-Gaussian processes. He is co-organizer and co-chair of the Eighth IEEE Signal Processing Workshop on Statistical Signal and Array Processing, Corfu, Greece (June 1996).

Jerry M. Mendel

Jerry Mendel received his B.S. degree in Mechanical Engineering in 1959, his M.S. in 1960, and his Ph.D. in 1963 in Electrical Engineering from the Polytechnic Institute of Brooklyn, NY. Currently he is Professor of Electrical Engineering at USC in Los Angeles.

Dr. Mendel is a Fellow of the IEEE, a Distinguished Member of the IEEE Control Systems Society, a member of Tau Beta Pi, Pi Tau Sigma, and Sigma Xi, and a registered Professional Control Systems Engineer in California. He has authored more than 300 technical papers, three textbooks and four other books related to his research in estimation theory, deconvolution, higher-order statistics, neural networks and fuzzy logic.

Chrysostomos L. (Max) Nikias

Chrysostomos L. (Max) Nikias received his B.S. degree in Electrical and Mechanical Engineering from the National Technical University of Athens, Greece and his M.S. and Ph.D. degrees in Electrical Engineering from the State University of New York at Buffalo in 1980 and 1982. Currently he is a Professor of Electrical Engineering at USC in Los Angeles, where he is also Director of CRASP and Associate Dean of Academic Research.

Dr. Nikias is a Fellow of the IEEE. He is the author of over 150 journal and conference papers, two textbooks, a monograph, and four patents. He is co-author of the textbook *Higher-Order Spectral Analysis: A Nonlinear Signal Processing Framework*, Prentice-Hall, Inc. 1993.

1

Tutorial

Introduction

This section of the *User's Guide* describes how to begin using the Higher-Order Spectral Analysis Toolbox for your signal processing applications. It assumes familiarity with basic MATLAB®, as well as a basic understanding of signals and systems.

There is much more information in a stochastic non-Gaussian or deterministic signal than is conveyed by its autocorrelation or power spectrum. Higher-order spectra, which are defined in terms of the higher-order moments or cumulants of a signal, contain this additional information. The Higher-Order Spectral Analysis (HOSA) Toolbox provides comprehensive higher-order spectral analysis capabilities for signal processing applications. The toolbox is an excellent resource for the advanced researcher and the practicing engineer, as well as the novice student who wants to learn about concepts and algorithms in statistical signal processing.

The Higher-Order Spectral Analysis Toolbox is a collection of M-files that implement a variety of advanced signal processing algorithms for spectral estimation, polyspectral estimation, and computation of time-frequency distributions, with applications such as parametric and nonparametric blind system identification, time delay estimation, harmonic retrieval, direction of arrival estimation, parameter estimation of Volterra (nonlinear) models, and adaptive linear prediction. Other potential applications include acoustics, biomedicine, econometrics, exploration seismology, nondestructive testing, oceanography, plasma physics, radar, sonar, speech etc.

For the newcomer to the field of higher-order statistics (spectra), some excellent starting places are:

[T2] Mendel, J.M., "Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications," *Proc. IEEE*, Vol. 79, pp. 278-305, 1991.

[T3] Nikias, C.L. and J.M. Mendel, "Signal processing with higher-order spectra," *IEEE Signal Processing Magazine*, Vol. 10, No 3, pp. 10-37, July 1993

[T4] Nikias, C.L. and A.P. Petropulu, *Higher-Order Spectra Analysis: A Nonlinear Signal Processing Framework*, New Jersey: Prentice-Hall, 1993.

[T1] Nikias, C.L. and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

The field of higher-order statistics (spectra) and its applications to various signal processing problems are relatively new. As such, there is no guarantee that a particular Higher-Order Spectral Analysis Toolbox routine will work well on your data. United Signals Systems, Inc., will be updating and upgrading the Higher-Order Spectral Analysis Toolbox from time to time to incorporate new routines and provide users with guidance on the applicability of existing routines as more experience is obtained through their use.

The "Tutorial" has numerous examples that reinforce the theory and demonstrate how to use the toolbox functions. All of the data files used by these examples are included in your Higher-Order Spectral Analysis Toolbox distribution diskette, and are described in the section on "Data Files." We encourage you to try out the examples yourself. Additional examples may be found in the demo, which can be invoked via `hosademo`. A later section in the "Tutorial" demonstrates how to deal with *real data*.

Polyspectra and Linear Processes

In this section, we will define cumulants, polyspectra, and various other related statistics, such as bicepstra and bicoherence. We will discuss tests for *linearity* and *Gaussianity*, and we will develop cumulant-based algorithms for estimating the parameters of linear (e.g., ARMA) processes.

Introduction

The notion of decomposing a signal into its harmonic components dates back to the analysis of the motion of planets ("the music of the spheres," as the Pythagorians called it), phases of the moon, laws of musical harmony, Newton's spectral decomposition of light (1677), Bernouilli (1738) and Euler's (1755) analysis of vibrating membranes, and Prony's approximation for vibrating mechanisms (1793). Modern Fourier Analysis, as we know it today, received its foundations in the work of Fourier (1807), although the roots of the Fast Fourier Transform (FFT) can be traced back to Gauss's work on orbital mechanics (1805).

We will assume, without loss of generality, that the processes or signals of interest to us are zero mean. We will also assume that the processes are discrete-time, with a sampling interval of $T = 1$, corresponding to a normalized sampling frequency of 1 Hz, so that the Nyquist frequency is 0.5 Hz.

The *power spectrum* is the primary tool of signal processing, and algorithms for estimating the power spectrum have found applications in areas such as radar, sonar, seismic, biomedical, communications, and speech signal processing. Analog equipment to estimate the spectrum, namely the *spectrum analyzer*, has been around for more than five decades, and may be found in almost any lab. Our toolbox not only offers a substitute for that equipment; it expands the analyst's toolkit to include algorithms more sophisticated than the simple conventional spectral analysis techniques.

The usefulness of the *power spectrum* arises from an important theorem, known as Wold's decomposition, which states that any discrete-time stationary random process can be expressed in the form,

$$x(n) = y(n) + z(n)$$

such that:

1 Processes $y(n)$ and $z(n)$ are uncorrelated with one another;

2 Process $y(n)$ has a causal linear process representation,

$$y(n) = \sum_{k=0}^{\infty} h(k)u(n-k)$$

where $h(0) = 1$, $\sum_{k=0}^{\infty} h^2(k) < \infty$, and $u(n)$ is a white-noise process; and,

3 $z(n)$ is *singular*, that is, it can be predicted perfectly (with zero variance) from its past.

An example of a singular process is the harmonic process, $s(n) = \alpha \exp(j2\pi fn)$. A process with $z(n) \equiv 0$ has a purely continuous spectrum; additionally, a strictly band-limited process is also singular.

Since real world signals cannot be strictly band limited, we may think of the Wold decomposition as decomposing a process into a linear process (which has a continuous spectrum) and a harmonic process (which has a line spectrum).

It is also important to note that the theorem only states that $u(t)$ is uncorrelated; it does not state that $u(t)$ is i.i.d., (higher-order white). For example, $u(n)$ might be the output of an all-pass system whose input is an i.i.d. process. We need higher-order statistics to determine whether or not $u(t)$ is i.i.d., or merely uncorrelated. Other motivations for using higher-order statistics (HOS) are discussed throughout this "Tutorial".

Definitions

The *autocorrelation* function or sequence of a stationary process, $x(n)$, is defined by,

$$R_{xx}(m) := E\{x^*(n)x(n+m)\} \tag{1-1}$$

where $E\{\cdot\}$ denotes the ensemble expectation operator. The power spectrum is formally defined as the Fourier Transform (FT) of the autocorrelation sequence (the Wiener-Khintchine theorem)

$$P_{xx}(f) = \sum_{m=-\infty}^{\infty} R_{xx}(m)\exp(-j2\pi f m) \tag{1-2}$$

where f denotes the frequency. An equivalent definition is given by

$$P_{xx}(f) := E\{X(f)X^*(f)\} \tag{1-3}$$

where $X(f)$ is the Fourier Transform of $x(n)$

$$X(f) = \sum_{n=-\infty}^{\infty} x(n)\exp(-j2\pi f n). \tag{1-4}$$

A sufficient, but not necessary, condition for the existence of the power spectrum is that the autocorrelation be absolutely summable. The power spectrum is real valued and nonnegative, that is, $P_{xx}(f) \geq 0$; if $x(n)$ is real valued, then the power spectrum is also symmetric, that is, $P_{xx}(f) = P_{xx}(-f)$.

As we shall see next, higher-order moments are natural generalizations of the autocorrelation, and cumulants are specific nonlinear combinations of these moments.

The first-order cumulant of a stationary process is the mean, $C_{1x} := E\{x(t)\}$. The higher-order cumulants are invariant to a shift of mean; hence, it is convenient to define them under the assumption of zero mean; if the process has nonzero mean, we subtract the mean, and then apply the following definitions to the resulting process. The second-, third- and fourth-order cumulants of a zero-mean stationary process are defined by [4],

$$C_{2x}(k) = E\{x^*(n)x(n+k)\} \tag{1-5}$$

$$C_{3x}(k,l) = E\{x^*(n)x(n+k)x(n+l)\} \tag{1-6}$$

$$\begin{aligned} C_{4x}(k,l,m) &= E\{x^*(n)x(n+k)x(n+l)x^*(n+m)\} \\ &\quad - C_{2x}(k)C_{2x}(l-m) - C_{2x}(l)C_{2x}(k-m) \\ &\quad - M^*_{2x}(m)M_{2x}(k-l) \end{aligned} \tag{1-7}$$

where $M_{2x}(m) = E\{x(n)\,x(n+m)\}$, and equals $C_{2x}(m)$, for a real-valued process. The first-order cumulant is the mean of the process; and the second-order cumulant is the autocovariance sequence. Note that for complex processes, there are several ways of defining cumulants depending upon which terms are conjugated.

The zero-lag cumulants have special names: $C_{2x}(0)$ is the variance and is usually denoted by σ_x^2; $C_{3x}(0,0)$ and $C_{4x}(0,0,0)$ are usually denoted by γ_{3x} and γ_{4x}. We will refer to the normalized quantities, $\gamma_{3x}/\sigma_{2x}^3$ as the *skewness* and $\gamma_{4x}/\sigma_{2x}^4$ as the *kurtosis*. These normalized quantities are both shift and scale invariant. If $x(n)$ is symmetric distributed, its *skewness* is necessarily zero (but not vice versa); if $x(n)$ is Gaussian distributed, its *kurtosis* is necessarily zero (but not vice versa). Often the terms skewness and kurtosis are used to refer to the unnormalized quantities, γ_{3x} and γ_{4x}.

If $x(n)$ is an i.i.d. process, its cumulants are nonzero only at the origin. If $x(n)$ is statistically independent of $y(n)$, and $z(n) = x(n) + y(n)$, then

$$C_{4z}(k,l,m) = C_{4x}(k,l,m) + C_{4y}(k,l,m),$$

with similar relationships holding for cumulants of all orders. This additivity property simplifies cumulant-based analysis.

The cumulants of a stationary real-valued process are symmetric in their arguments, that is,

$$C_{2x}(k) = C_{2x}(-k)$$

$$C_{3x}(k, l) = C_{3x}(l, k) = C_{3x}(-k, l-k)$$

$$C_{4x}(k, l, m) = C_{4x}(l, k, m) = C_{4x}(k, m, l) = C_{4x}(-k, l-k, m-k)$$

Hence, the fundamental region of support is not the entire k-D plane. For example, for $k = 2$, $C_{2x}(k)$, $k \geq 0$, specifies $C_{2x}(k)$ everywhere. It is easily shown that the nonredundant region for $C_{3x}(k,l)$ is the wedge

$$\{(k,l) : 0 \leq l \leq k \leq \infty\},$$

and for $C_{4x}(k,l,m)$, it is the cone,

$$\{(k,l,m) : 0 \leq m \leq l \leq k \leq \infty\}.$$

The kth-order polyspectrum is defined as the FTs of the corresponding cumulant sequence:

$$S_{2x}(f) = \sum_{k=-\infty}^{\infty} C_{2x}(k) e^{-j2\pi f \kappa} \tag{1-8}$$

$$S_{3x}(f_1, f_2) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} C_{3x}(k, l) e^{-j2\pi f_1 k} e^{-j2\pi f_2 l} \tag{1-9}$$

$$S_{4x}(f_1, f_2, f_3) = \sum_{k,\, l,\, m=-\infty}^{\infty} C_{4x}(k, l, m) e^{-j2\pi(f_1 k + f_2 l + f_3 m)} \tag{1-10}$$

which are respectively the power spectrum, the bispectrum, and the trispectrum. Note that the bispectrum is a function of two frequencies, whereas the trispectrum is a function of three frequencies. In contrast with the power spectrum which is real valued and nonnegative, bispectra and trispectra are complex valued.

For a real-valued process, symmetry properties of cumulants carry over to symmetry properties of polyspectra. The power spectrum is symmetric: $S_{2x}(f) = S_{2x}(-f)$. The symmetry properties of the bispectrum are given by [56]:

$$\begin{aligned} S_{3x}(f_1, f_2) &= S_{3x}(f_2, f_1) = S_{3x}(f_1, -f_1 - f_2) \\ &= S_{3x}(-f_1, -f_2, f_2) = S^*_{3x}(-f_1, -f_2). \end{aligned} \tag{1-11}$$

Hence, a nonredundant region of support for the bispectrum is the triangle with vertices (0,0), (1/3,1/3) and (1/2,0); recall that we have assumed a normalized sampling frequency of 1 Hz.

Symmetry properties of the trispectrum include:

$$\begin{aligned} S_{4x}(f_1, f_2, f_3) &= S_{4x}(f_1, f_3, f_2) = S_{4x}(f_2, f_1, f_3) \\ &= S_{4x}(-f_1, f_2 - f_1, f_3 - f_1) = S^*_{4x}(-f_1, -f_2, -f_3). \end{aligned}$$

The literature is somewhat confusing both in the derivation, as well as in the description of the nonredundant regions. The nonredundant region for a continuous-time band-limited process, with a Nyquist frequency of 0.5 Hz, is the triangle with vertices (0,0), (1/4,1/4), (1/2,0). A tutorial treatment of the differences between the continuous-time and the discrete-time cases is given in [47]; related discussions may be found in [63]. The nonredundant region of the trispectrum is discussed in [6, 9, 47].

Similar to the cross-correlation, we can also define cross-cumulants; for example,

$$C_{xyz}(k, l) = E\{x^*(n)y(n + k)z(n + l)\} \tag{1-12}$$

The cross-bispectrum is defined by,

$$S_{xyz}(f_1, f_2) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} C_{xyz}(k, l) e^{-j2\pi f_1 k} e^{-j2\pi f_2 l} \tag{1-13}$$

Note that the bispectrum $S_{3x}(f_1,f_2)$ is a special case of the cross-bispectrum obtained when $x = y = z$.

The *cross-bicoherence* is another useful statistic which is defined as,

$$\mathrm{bic}_{xyz}(f_1, f_2) = \frac{S_{xyz}(f_1, f_2)}{\sqrt{S_{2x}(f_1 + f_2) S_{2y}(f_1) S_{2z}(f_2)}} \tag{1-14}$$

The autobicoherence is obtained when $x = y = z$. M-file `bicoherx` can be used to estimate the cross-bicoherence, and `bicoher` can be used to estimate the bicoherence.

The *cross-bicepstrum* of three processes is defined by

$$b_{xyz}(m, n) = \int \int \ln(S_{xyz}(f_1, f_2)) e^{j2\pi f_1 m} e^{j2\pi f_2 n} df_1 df_2 \tag{1-15}$$

and is well-defined only if $S_{\mathrm{xyz}}(f_1,f_2)$ is nonzero everywhere.

Why Do We Need Higher-Order Statistics?

Motivation to use cumulants and polyspectra of order $k > 2$ is given by the following ($\mathbf{m}_k = (m_1 \ldots, m_{k-1})$):

- If $z(n) = x(n) + y(n)$, and $x(n)$ and $y(n)$ are mutually independent processes, then $C_{kz}(\mathbf{m}_k) = C_{kx}(\mathbf{m}_k) + C_{ky}(\mathbf{m}_k)$.
- If $x(n)$ is Gaussian, then $C_{kz}(\mathbf{m}_k) = 0$, $k > 2$.
- Hence, if $z(n) = x(n) + w(n)$, where $w(n)$ is Gaussian and independent of $x(n)$, then, for $k > 2$, $C_{kz}(\mathbf{m}_k) = C_{kx}(\mathbf{m}_k)$. Thus, we can recover the higher-order cumulants of a non-Gaussian signal even in the presence of colored Gaussian noise.
- Let $x(n)$ be a linear process, that is, $x(n) = \sum_k h(k)u(n-k)$, where $u(n)$ is i.i.d. Then, it follows that:

$$C_{2x}(k) = \gamma_{2u} \sum_n h^*(n)h(n+k) \tag{1-16}$$

$$C_{3x}(k, l) = \gamma_{3u} \sum_n h^*(n)h(n+k)h(n+l) \tag{1-17}$$

$$C_{4x}(k, l, m) = \gamma_{4u} \sum_n h^*(n)h(n+k)h(n+l)h^*(n+m) \tag{1-18}$$

$$S_{2x}(f) = \gamma_{2u}|H(f)|^2 \tag{1-19}$$

$$S_{3x}(f_1, f_2) = \gamma_{3u}H(f_1)H(f_2)H^*(f_1 + f_2) \tag{1-20}$$

$$S_{4x}(f_1, f_2, f_3) = \gamma_{4u}H(f_1)H(f_2)H(f_3)H^*(f_1 + f_2 + f_3) \tag{1-21}$$

where $\gamma_{ku} = C_{ku}(0)$. Note that the power spectrum does not carry any information about the phase of $H(f)$. In contrast, if $u(n)$ is non-Gaussian, this phase information can be recovered from the higher-order polyspectra. Thus, the standard *minimum-phase* assumption, which is necessary when the process is Gaussian or only second-order statistics are used, may be dropped.

- Any process can always be considered to be a linear process with respect to its second-order statistics; that is, given R_{yy}, we can always find $\{h(k)\}$ and an uncorrelated process $u(n)$, such that, $R_{yy}(m) = R_{xx}(m)$, where $x(n) = \sum_k h(k)u(n-k)$. In other words, the autocorrelation sequence cannot give any evidence of nonlinearity. In contrast, higher-order cumulants can give evidence of nonlinearity.
- Processes of the form $x(t) = a(t)\exp(j\sum_{k=0}^{p} a_k t^k)$ whose phase is a polynomial in time t, are called *polynomial phase* processes; the FTs of such processes tend to be flat, whereas suitably defined slices of higher-order spectra reveal structure that permits estimation of p and the a_k's.

To summarize, cumulants are useful: (1) if the additive noise is Gaussian and the signal is non-Gaussian, (2) the linear system is non-minimum phase (that is, mixed-phase), or (3) the process is nonlinear.

Bias and Variance of an Estimator

In practice, we estimate cumulants and polyspectra from data. These estimates are, themselves, random, and are characterized by their *bias* and *variance*.

Let $x(n)$ denote a stationary process; we assume that all relevant statistics exist and have finite values. Let s denote some statistic, defined on $x(n)$. Let $\hat{s}_N$ denote an estimate of the statistic based on N observations, $\{x(n)\}_{n=0}^{N-1}$. Since $x(n)$ is a random process, the estimate $\hat{s}_N$ is also random; clearly, $\hat{s}_N$ will not

equal s. The estimate $\hat{s}_N$ is a good estimate if it is "near" s. This notion is clarified by introducing the ideas of *bias* and *consistency*.

The bias of an estimator is defined as $E\{\hat{s}_N\} - s$; the estimate is said to be unbiased if the bias is zero, that is,

$$E\{\hat{s}_N\} = s.$$

Often this holds true only as $N \to \infty$, in which case the estimate is said to be *asymptotically unbiased*.

The bias, by itself, does not completely characterize the estimate. If the estimate is good, we expect that $\hat{s}_N$ will take on values around the true quantity s. The natural measure of the spread is the squared deviation around the true quantity, s,

$$E\left\{\left|\hat{s}_N - s\right|^2\right\}.$$

The estimate is said to be (asymptotically) consistent if the squared deviation goes to zero, as $N \to \infty$. This condition is sometimes called *mean-square consistency*. A consistent estimate is necessarily (asymptotically) unbiased.

Estimating Cumulants

In practice, we have a finite amount of data, $\{x(n)\}_{n=0}^{N-1}$, and we must obtain consistent estimates of cumulants. The sample estimates are given by,

$$\hat{C}_{xy}(k) = \frac{1}{N_3} \sum_{n=N_1}^{N_2} x^*(n)y(n+k) \tag{1-22}$$

$$\hat{M}_{xy}(k) = \frac{1}{N_3} \sum_{n=N_1}^{N_2} x(n)y(n+k) \tag{1-23}$$

$$\hat{C}_{xyz}(k,l) = \frac{1}{N_3} \sum_{n=N_1}^{N_2} x^*(n)y(n+k)z(n+l) \tag{1-24}$$

$$\hat{C}_{wxyz}(k, l, m) = \frac{1}{N_3} \sum_{n=N_1}^{N_2} w^*(n)x(n+k)y(n+l)z^*(n+m)$$
$$- \hat{C}_{wx}(k)C_{yz}(l-m) - \hat{C}_{wy}(l)C_{xz}(k-m)$$
$$- \hat{M}^*_{wz}(m)\hat{M}_{xy}(l-k) \qquad \textbf{(1-25)}$$

where N_1 and N_2 are chosen such that the summations involve only $x(n)$'s with $n \in [0, N-1]$; unbiased estimates are obtained if N_3 is set equal to the actual number of terms which are averaged; for example,

$$E\{\hat{C}^N_{xyz}(k, l)\} = C_{xyz}(k, l)$$

Usually we set N_3 to N and obtain estimates that are asymptotically unbiased. Autocumulants are obtained when $w = x = y = z$. These estimates are known to be consistent provided the process $x(n)$ satisfies some weak mixing conditions [5]. For example, for large N, the variance of the sample estimate of the third-order cross-cumulant is given by

$$\text{var}\{\hat{C}^N_{xyz}(k, l)\} = c/N$$

where c is a finite constant that depends upon the auto- and cross-moments (cumulants) of orders 1 through 6 of the processes $x(n)$, $y(n)$, and $z(n)$.

These definitions assume that the processes are zero mean; in practice, the sample mean is removed first. Routines `cum2x`, `cum3x`, and `cum4x` may be used to estimate cross-cumulants of orders 2, 3, and 4; `cumest` may be used to estimate the autocumulants.

Examples

We will simulate a non-Gaussian ARMA process, and then estimate its cumulants:

```
rand('seed',0); randn('seed',0);
u=rpiid(1024,'exp'); n=25;
y=filter([1,-2], [1,-1.5,0.8], u);
for k=-n:n,
cmat(:,k+n+1)=cumest(y,3,n,128,0,'biased',k);
end
subplot(121), mesh(-n:n,-n:n, cmat)
subplot(122), contour(-n:n,-n:n,cmat,8)
```

Time-series **y** is segmented into records of 128 samples each, with no overlap; biased estimates of the third-order cumulants are obtained from each segment and then averaged; the (*i*,*j*) element of `cmat` will contain the estimate of $C_{3y}(i-n-1, j-n-1)$, for $i,j = 1, \ldots, 2*n+1$. You can use the function `cumtrue` to compute and display the true cumulants.

The contour plot in Figure 1-1 reveals the basic symmetry of third-order cumulants, namely $C_{3y}(\tau_1,\tau_2) = C_{3y}(\tau_2,\tau_1)$. Other symmetry properties may be verified by using `cumtrue` to estimate the true cumulants of a linear process.

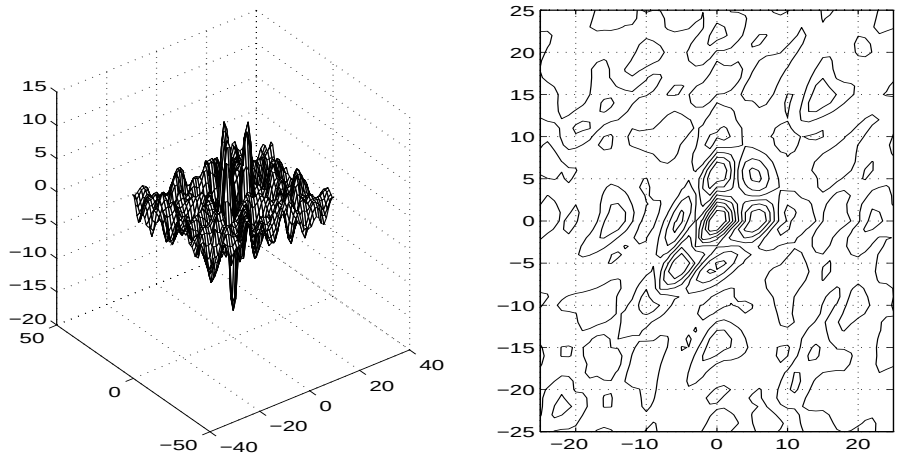


Figure 1-1: Estimated Third-Order Cumulants of an ARMA(2,1) Process (cumest)

Estimating Polyspectra and Cross-polyspectra

Estimators of polyspectra are natural extensions of estimators of the power spectrum, with some important differences in the smoothing requirements. Hence, it will be useful to review power spectrum estimation techniques first.

Estimating the Power Spectrum

Techniques for estimating the conventional power spectrum fall into three broad categories: the nonparametric or conventional methods, the parametric or model-based methods, and the criterion-based methods.

The first category includes two classes: the direct methods, which are based on the FT of the observed data; and indirect methods, which are based on computing the FT of the estimated autocorrelation sequence of the data. The class of parametric methods includes algorithms such as MA, AR, and ARMA modeling, and eigen-space based methods such as MUSIC, Min-Norm, etc., which are appropriate for harmonic models. Criterion-based methods include Burg's Maximum Entropy algorithm and Capon's Maximum-Likelihood algorithm.

The conventional estimators are easy to understand and easy to implement, but are limited by their resolving power (the ability to separate two closely spaced harmonics), particularly when the number of samples is small. For random signals, these estimators typically require long observation intervals in order to achieve acceptably low values for the variances of the estimate.

The natural estimator of the power spectrum is the FT of $\hat{R}_{xx}^{N}(m)$,

$$I_{xx}^{N}(f) = \sum_{m=-N-1}^{N-1} \hat{R}_{xx}^{N}(m) e^{-j2\pi fm} = \frac{1}{N}\left|\sum_{k=0}^{N-1} x(k) e^{-j2\pi fk}\right|^2 .$$

This estimator, also known as the periodogram, can be computed as the squared magnitude of an N-point FFT of the observed time series. Since $E\{\hat{R}_{xx}^{N}(m)\} = R_{xx(m)}$, the periodogram is an unbiased estimator of $P_{xx}(f)$. However, the periodogram is not a consistent estimator, because $\mathrm{var}(P_{xx}^{N}(f)) = P_{xx}^{2}(f)$; that is, its variance does not go to zero as $N \to \infty$. [56]

If $x(n)$ is Gaussian white noise with variance σ^2, its power spectrum should be flat, $P_{xx}(f) = \sigma^2$; the variance of the periodogram estimate, $P_{xx}^N(f)$, is given by,

$$\mathrm{var}(P_{xx}^N(f)) = \sigma^4\left[1 + \left(\frac{\sin(2\pi f N)}{N\sin(2\pi f)}\right)^2\right].$$

Note that the variance does not go to zero as $N \to \infty$, that is, the estimate is not consistent. The covariance between estimates at frequencies $f_1 = k/N$ and $f_2 = m/N$ is zero; thus, on the one hand as N increases, the variance at any f does not go to zero; however, the spacing between estimates that are uncorrelated decreases as $1/N$; as a consequence, the fluctuations in the periodogram become more rapid as N increases. As an example, try `semilogy(abs(fft(rpiid(n,'nor'))))` for increasing values of n. Proper smoothing smooths out the fluctuations and yields consistent estimates.

It should be emphasized that the variance expressions are meaningful only in the context of random processes; the FT, itself, is a very useful tool for analyzing deterministic signals.

The periodogram estimate can be made consistent in several ways, by:

- Smoothing (filtering) in the frequency domain;
- Multiplying the autocorrelation sequence by a lag window function;
- Multiplying the time-domain data by a window function; or,
- Averaging several periodogram estimates.

These observations carry over to bispectral estimates as we see next.

Estimating Bispectra and Cross-Bispectra

The natural estimate of the cross-bispectrum is the FT of the third-order cumulant sequence, that is,

$$I_{xyz}^N(f) = \sum_{k=-N-1}^{N-1}\sum_{l=-N-1}^{N-1} \hat{C}_{xyz}(k,l)e^{-j2\pi f_1 k}e^{-j2\pi f_2 l}$$

$$= \frac{1}{N^2}X_N^*(f_1+f_2)Y_N(f_1)Z_N(f_2) \qquad \textbf{(1-26)}$$

where $X_N(f)$ is the FT of $\{x(n)\}_{n=0}^{\cdot\cdot}$. This estimate, known as the *cross-biperiodogram*, is not a consistent estimate. As in the case of the power spectrum, the estimate can be made consistent by suitable smoothing. The bispectrum and the biperiodograms are special cases obtained when $x = y = z$.

Smoothing can be accomplished by multiplying the third-order cumulant estimates by a lag window function. Let $w(t,s)$ be a 2-D window function, whose 2-D FT is bounded and nonnegative; further, assume

$$w(0,0) = 1; \qquad \int\int w^2(t,s)\,dt\,ds < \infty;$$
$$\int\int f_i^2 W(f_1,f_2)\,df_1\,df_2 < \infty; \qquad \int\int f_i W(f_1,f_2)\,df_1\,df_2 = 0;$$

The window function, $w(t,s)$, must also satisfy the symmetry properties of third-order cumulants. For example, 2-D lag windows may be derived from 1-D lag windows as follows,

$$w(t,s) = w(t)w(s)w(t-s)$$

which satisfies the symmetry conditions of $C_{3x}(m,n)$.

Consider the scaled-parameter window, $w_M(t,s) = w(t/M, s/M)$, and the smoothed estimate,

$$\hat{S}_{xyz}(f_1,f_2) = \sum_{k=-N-1}^{N-1} \sum_{l=-N-1}^{N-1} \hat{C}_{xyz}(k,l) w_M(k,l) e^{-j2\pi f_1 k} e^{-j2\pi f_2 l} \qquad \textbf{(1-27)}$$

Under the assumption that the cross-bispectrum $S_{xyz}(f_1,f_2)$ is sufficiently smooth, the smoothed estimate is known to be consistent, with variance given by,

$$\mathrm{var}(\hat{S}_{xyz}(f_1,f_2)) = \frac{M^2}{N} S_{2x}(f_1+f_2) S_{2y}(f_1) S_{2z}(f_2) \int\int w^2(t,s)\,dt\,ds \qquad \textbf{(1-28)}$$

for $0 < f_1 < f_2 < \pi$. Note the implied consistency condition is $M \to \infty$ and $M^2/N \to \infty$, as $N \to \infty$, and $\int\int w^2(t,s)dt\,ds < \infty$. The estimator in (1-27), for $x = y = z$, is implemented in routine `bispeci`.

An alternative approach is to perform the smoothing in the frequency domain. As in the case of power spectra, we may segment the data into K records of

length $L = N/K$, compute and average the biperiodograms, and then perform the frequency smoothing, using the frequency-domain filter, $W_M(f_1,f_2)$, the FT of $w_M(t,s)$. In this case,

$$\mathrm{var}(\hat{S}_{xyz}(f_1, f_2)) = \frac{M^2}{LK} S_{2x}(f_1 + f_2) S_{2y}(f_1) S_{2z}(f_2) \int\int w^2(t, s)\, dt\, ds \qquad \textbf{(1-29)}$$

for $0 < f_1 < f_2 < \pi$. Windowing is not required in this case provided K is large; however, this does not ensure that the bias will go to zero. Rao and Gabr [56, Sec. 2.4] have derived a bispectral window which is optimum in terms of bias and variance; windows satisfying other optimality criteria are discussed in [56, 40]. The Rao-Gabr window is available as an option in routine `bispeci`. Routine `bispecdx` computes estimates of the cross-bispectrum.

Examples

```
load qpc
bspec=bispeci(zmat,21,64,0,'unbiased',128,1);
dbspec=bispecdx(zmat,zmat,zmat,128,3,64,0);
```

The contour plots of the two estimates of the bispectrum are shown in Figure 1-2 and Figure 1-3.

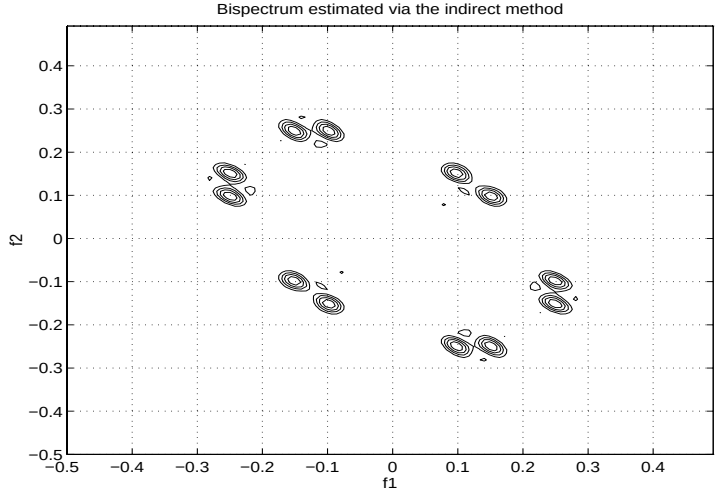


Figure 1-2 Indirect Estimate of the Bispectrum (bispeci)

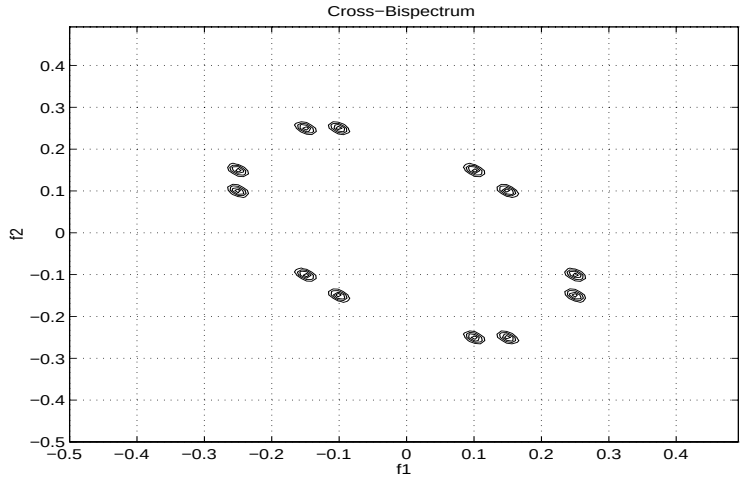


Figure 1-3 Direct Estimate of the Bispectrum (bispecdx)

Both the direct and the indirect estimates reveal peaks at (0.10, 0.15) and the 11 other symmetric locations as indicated by (1.11). The data used in this example consist of quadratically phase-coupled harmonics with frequencies at 0.10, 0.15, and 0.25 Hz, and an uncoupled harmonic at 0.40 Hz; quadratic phase-coupling is discussed further in the section on "Nonlinear Processes," where we will see that the presence of pronounced peaks in the bispectrum is indicative of nonlinear phenomena.

Let $x(n)$ be a (zero mean) white Gaussian process, with variance σ_x^2; let $y(n) = x^2(n)$. It is easy to show that the cross-bispectrum of (x,x,y) should be a constant, equal to $2\sigma_x^4$. This follows by noting that if a,b,c,d are zero mean and jointly Gaussian, then,

$$E\{abcd\} = cum(a,b)cum(c,d) + cum(a,d)cum(b,c) + cum(a,c)cum(b,d).$$

Examples

```
randn('seed',0);
x=randn(64,64); y=x.*x;
dbic=bispecdx(x,x,y,128,5);
```

Notice that an apparent structure along the axes (which is an artifact due to the removal of the mean), consistent estimates along the axes, and the

anti-diagonal can be obtained only by sufficient smoothing; a smoothing window of size 5 is inadequate. The lines $\omega_1 = 0$, $\omega_2 = 0$, and $\omega_1 + \omega_2 = 0$, are called the principal submanifolds, [4, 6]. This example also illustrates that passing a Gaussian process through a nonlinearity makes it non-Gaussian.

Estimating Bicoherence

Given estimates of the power spectra and the cross-bispectrum, we can estimate the cross-bicoherence as indicated in (1-14). It has been shown that consistent estimates of the power spectrum and the bispectrum lead to consistent estimates of the bicoherence.

Routines `bicoher` and `bicoherx` may be used to estimate the autobicoherence and the cross-bicoherence.

Examples

```
load qpc
dbspec=bicoher(zmat,128);
```

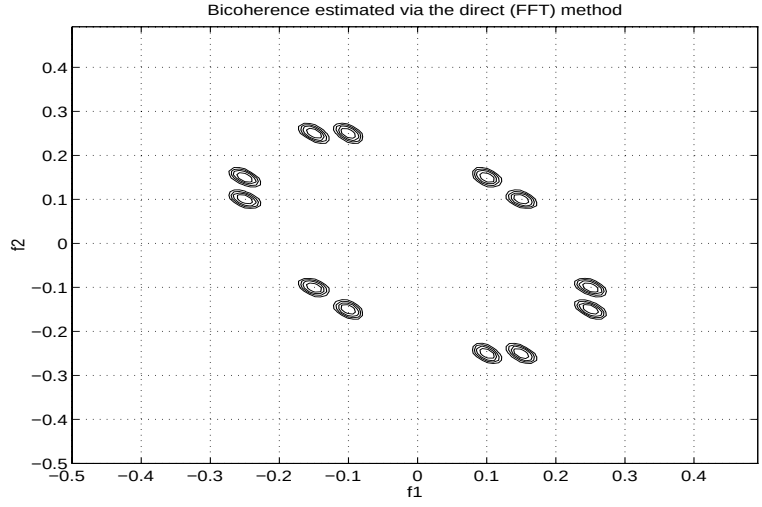


Figure 1-4 Bicoherence Estimate (bicoher)

A contour plot of the estimated bicoherence is shown in Figure 1-4. The data consists of quadratically phase-coupled harmonics with frequencies at 0.10,

0.15, and 0.25 Hz, and an uncoupled harmonic at 0.40 Hz. The maximum value of the bicoherence is 0.6442; the value is less than unity because of the additive noise, which affects the power spectrum estimate.

```
load nl1
bicx=bicoherx(x,x,y);
```

You should see the display in Figure 1-5. The cross-bicoherence is significantly nonzero, and nonconstant, indicating a nonlinear relationship between `x` and `y`. The nonsharpness of the peaks, as well as the presence of structure around the origin, indicates that the nonlinear relationship is not purely of the form $y(n) = x^2(n)$, and that x,y are not narrow-band processes. From the description of `nl1.mat`, we see that `y` is the output of a second-order Volterra system whose input, `x`, is Gaussian.

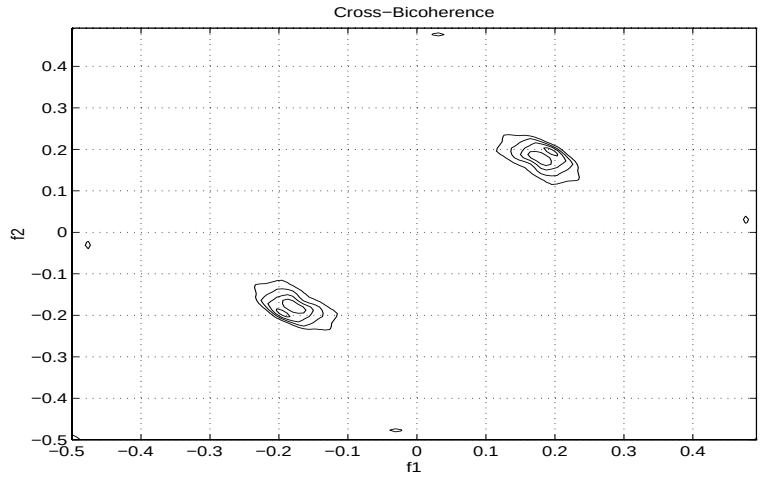


Figure 1-5 Cross-Bicoherence Estimate (bicoherx)

Testing for Linearity and Gaussianity

In subsequent sections we discuss algorithms for the estimation of parameters of non-Gaussian linear processes; of course, these routines assume that the data are linear and non-Gaussian. We call a process, $y(n)$, linear, if it can be represented by

$$y(n) = \sum_k h(k)u(n-k) \ ,$$

where $u(n)$ is assumed to be i.i.d. If $u(n)$ is Gaussian (non-Gaussian), we say that $y(n)$ is linear Gaussian (non-Gaussian). How do we know that the data are non-Gaussian, and that they are additionally linear?

Hinich [24] has developed algorithms to test for non-skewness (loosely called Gaussianity) and linearity. The basic idea is that if the third-order cumulants of a process are zero, then its bispectrum is zero, and hence its bicoherence is also zero. If the bispectrum is not zero, then the process is non-Gaussian; if the process is linear and non-Gaussian, then the bicoherence is a nonzero constant: see (1-19), (1-20), and (1-14) with $x = y = z$. Thus, we have a hypothesis testing problem for non-Gaussianity (non-zero bispectrum):

H1: the bispectrum of $y(n)$ is nonzero;

H0: the bispectrum of $y(n)$ is zero.

If hypothesis **H1** holds, we can test for linearity, that is, we have a second hypothesis testing problem,

H1': the bicoherence of $y(n)$ is not constant;

H0': the bicoherence of $y(n)$ is a constant.

If hypothesis **H0'** holds, the process is linear.

Assume that we have good (perfect) estimates of the power spectrum, and consider the sample estimate of the squared bicoherence,

$$\left|\hat{\text{bic}}_{xxx}(f_1, f_2)\right|^2 = \frac{\left|\hat{S}_{xxx}(f_1, f_2)\right|^2}{S_{2x}(f_1 + f_2)S_{2x}(f_1)S_{2x}(f_2)} \qquad \textbf{(1-30)}$$

It has been established in [6] that sample estimates of the bispectrum using conventional methods are asymptotically (complex) Gaussian; additionally, the estimates at different frequencies are uncorrelated, provided the frequency separation is greater than the reciprocal of the effective window length. If $\hat{S}_{xxx}$ is Gaussian distributed, we know that $|\hat{S}_{xxx}|^2$ is chi-squared distributed with two degrees of freedom.

If $\hat{S}_{xxx}(f_1, f_2) \equiv 0$, then the statistic in(1-30) is a central chi-squared r.v. with two degrees of freedom. The squared bicoherence is summed over the P points in the nonredundant region; details are given in [24]. The resulting statistic S is χ^2 distributed, with $2P$ degrees of freedom. Hence it is easy to devise a statistical test to determine whether the observed S is consistent with a central chi-squared distribution; this "consistency" is reported as probability-of-false alarm value, that is, the probability that we will be wrong in assuming that the data have a nonzero bispectrum. If this probability is small, say 0.95, we accept the assumption of zero bispectrum, that is, we cannot reject the Gaussianity assumption. This test is implemented in routine `glstat`.

Let us assume that we have estimated S and are confident that the data is non-Gaussian. Now if the data are also linear, we expect the squared bicoherence to be constant for all f_1 and f_2. In practice, the estimated bicoherence will not be flat: we can obtain an estimate of the constant value by computing the mean value of the bicoherence over the points in the nonredundant region; let λ denote this mean value. The squared bicoherence is chi-squared distributed with two degrees of freedom and noncentrality parameter λ. The sample interquartile range, R, of the squared bicoherence can be estimated, and compared with the theoretical interquartile range of a chi-squared distribution with two degrees of freedom and noncentrality parameter λ. If the estimated interquartile range is much larger or much smaller than the theoretical value, then we should reject the linearity hypothesis. This test is implemented in routine `glstat`.

Note that a zero bispectrum is not proof of Gaussianity, since the higher-order cumulants and polyspectra need not be identically zero. The bispectrum of a *time-reversible* process is identically equal to zero. For example, consider the linear process $x(n) = \sum_k h(k)u(n-k)$, where $u(n)$ is an i.i.d. process. If $u(n)$ is symmetrically distributed, then the bispectrum of $x(n)$ is zero. If $u(n)$ is Gaussian, the bispectrum as well as all higher-order polyspectra of $x(n)$ are identically zero. If $u(n)$ is Laplace distributed, the bispectrum and all odd-ordered polyspectra of $x(n)$ are zero, but the even-ordered polyspectra (such as the trispectrum) are not identically equal to zero.

Note also that univariate symmetry or zero skewness does not mean zero bispectrum. In the above example, suppose that $\gamma_{3u} \neq 0$, and $\sum_k h^3(k) = 0$; then, $C_{3x}(0,0) = \gamma_{3u} \sum_k h^3(k) = 0$, so that process $x(n)$ has zero skewness. This does not imply that $C_{3x}(\tau_1,\tau_2) \equiv 0$ or $S_{3x}(\omega_1,\omega_2) \equiv 0$. In particular, let $h(0) = -h(1) = 1$, $h(k) = 0$, $\forall k > 1$; then, $x(n) = u(n) - u(n-1)$ is easily seen to be symmetrically distributed around zero, but the bispectrum of $x(n)$ is not identically zero.

In general, we use notions of skew and symmetry in the context of random variables (or univariate pdfs); bispectra and trispectra relate to random processes (multivariate pdfs).

Examples

```
load gldat
glstat(g,0.51,256);
```

Test statistic for Gaussianity is 22.179 with df = 48, Pfa = 0.9995.

Linearity test: R (estimated) = 0.88819, lambda = 0.6982, R (theory) = 2.9288, N = 14.

The data in **g** are Gaussian distributed. Since the Pfa is high, we cannot reject the *Gaussian hypothesis*; but if the Gaussian hypothesis holds, the bispectrum must be zero, and we cannot conclude, on the basis of the bispectrum alone, whether or not the process is linear; hence, the results of the linearity test should be ignored in this case. In this case, the data are Gaussian, hence, also linear.

```
glstat(u,0.51,256);
```

Test statistic for Gaussianity is 17.4885 with df = 48, Pfa = 1.

Linearity test: R (estimated) = 0.72383, lambda = 0.51704, R (theory) = 2.7453, N = 14.

The data in **u** are uniformly distributed. Since the Pfa is high, we cannot reject the Gaussian hypothesis; but if the Gaussian hypothesis holds, the bispectrum must be zero, and we cannot conclude, on the basis of the bispectrum alone, whether or not the process is linear; hence, the results of the linearity test should be ignored in this case.

```
glstat(e,0.51,256);
```

Test statistic for Gaussianity is 253.3529 with df = 48, Pfa = 0.

Linearity test: R (estimated) = 7.8894, lambda = 9.4555, R (theory) = 8.4655, N = 14.

The data in **e** obey the single-sided exponential distribution. The Pfa in rejecting the Gaussian hypothesis is very small; hence, we are comfortable in accepting the hypothesis of non-Gaussianity. The estimated and theoretical interquartile ranges (the R's) are fairly close to one another; hence, we accept the linearity hypothesis. The estimate of the interquartile range was based on N=14 samples.

```
glstat(x,0.51,256);
```

Test statistic for Gaussianity is 277.5194 with df = 48, Pfa = 0.

Linearity test: R (estimated) = 6.7513, lambda = 10.6519, R (theory) = 8.968, N = 14.

x was obtained by passing through a linear filter; hence, it is linear and non-Gaussian. We reject the Gaussianity assumption since the Pfa is small; we accept the linearity hypothesis since the estimated and theoretical interquartile ranges are close to one another.

```
glstat(z,0.51,256);
```

Test statistic for Gaussianity is 12640.0657 with df = 48, Pfa = 0.

Linearity test: R (estimated) = 606.9323, lambda = 492.5759, R (theory) = 59.9088, N = 14.

z was obtained by passing x through a nonlinearity, $\mathbf{z(n) = x^3(n)}$: hence, **z** is non-Gaussian (and nonsymmetric). We reject the Gaussianity assumption since the Pfa is small; we cannot accept the linearity hypothesis since the estimated interquartile ranges is much larger than the theoretical value.

```
glstat(l,0.51,256);
```

Test statistic for Gaussianity is 49.931 with df = 48, Pfa = 0.3965.

Linearity test: R (estimated) = 2.6047, lambda = 1.8124, R (theory) = 4.0038, N = 14.

The data in **l** are i.i.d. and Laplace distributed (symmetric). Since the Pfa is high, we cannot reject the Gaussian hypothesis; but if the Gaussian hypothesis holds, the bispectrum must be zero, and we cannot conclude, on the basis of the bispectrum alone, whether or not the process is linear; hence, the results of the linearity test should be ignored in this case.

Parametric Estimators, ARMA Models

So far we have looked at nonparametric estimators. Parametric estimators are often useful, either because they lead to parsimonious estimates, or because the underlying physics of the problem suggest a parametric model.

The basic idea is that if $x(n)$ depends upon a finite set of parameters, θ, then all of its statistics can be expressed in terms of θ. For example, we obtain parametric estimates of the power spectrum by first estimating θ, and then evaluating $P_{xx}(f \mid \theta)$.

The specific form we postulate for the relationship between θ and the sequence $x(n)$ constitutes a model. A popular model in time-series analysis is the Auto-Regressive Moving-Average (ARMA) model,

$$x(n) = -\sum_{k=1}^{p} a(k)x(n-k) + \sum_{k=0}^{q} b(k)u(n-k) \qquad \textbf{(1-31)}$$

where $u(n)$ is assumed to be an i.i.d. sequence, with variance σ_u^2. The Auto-Regressive (AR) polynomial is defined by,

$$A(z) = \sum_{k=0}^{p} a(k)z^{-k} \qquad \textbf{(1-32)}$$

where $a(0) = 1$. $A(z)$ is assumed to have all its roots inside the unit circle, that is, $A(z_o) = 0 \rightarrow |z_o| < 1$; this condition is also referred to as the *minimum-phase* or *causal and stable* condition. In general, no restrictions need to be placed on the zeros of the Moving-Average (MA) polynomial,

$$B(z) = \sum_{k=0}^{q} b(k)z^{-k} \qquad \textbf{(1-33)}$$

however, $B(z)$ is usually assumed to be minimum-phase. The minimum-phase assumption is usually not true for discrete-time processes which are obtained by sampling a continuous-time process; algorithms based on HOS do not require this assumption.

The power-spectrum of the ARMA process is given by,

$$P_{xx}(f) = \sigma_u^2 \frac{|B(z)|^2}{|A(z)|^2} \qquad z = e^{j2\pi f} \tag{1-34}$$

Note that the power spectrum does not retain any phase information about the transfer function $H(z) = B(z)/A(z)$. Since we do not have access to the sequence $u(n)$, one either assumes that $u(n)$ has unit variance, or that $b(0) = 1$. Instead of estimating $P_{xx}(f) \forall f \in [-1/2, 1/2]$, as in the nonparametric approach, we have to estimate only $(p + q + 1)$ parameters, namely, $\{a(k)\}_{k=1}^{p}$, $\{b(k)\}_{k=1}^{q}$, and σ_u^2.

Let $h(n)$ denote the impulse response of the model in (1-31); hence, $H(z) = B(z)/A(z)$. The AR and MA parameters are related to the impulse response (IR) via,

$$\sum_{k=0}^{n} a(k)h(n-k) = b(n), \qquad n \in [0, q] \tag{1-35}$$

$$= 0, \qquad n \notin [0, q] \tag{1-36}$$

In practice, the observed process is noisy, that is,

$$y(n) = x(n) + w(n) \tag{1-37}$$

where process $w(n)$ is additive colored Gaussian noise; the color of the noise is usually not known.

Given the noisy observed data, $y(n)$, we want to estimate the $a(k)$'s and the $b(k)$'s in (1-31). We assume that the model orders p and q are known.

The determination of the model orders p and q is an important issue, and is discussed later (routines `arorder` and `maorder` implement AR and MA model order determination techniques). In general, it is better to overestimate model

orders rather than to underestimate them (however, specific implementations or algorithms may suffer from zero-divided-by-zero type problems).

Motivation to use cumulants for the ARMA parameter estimation problem is as follows:

- Correlation-based methods can be used successfully only if $q = 0$ (pure AR) and the additive noise is white.
- Even in the noiseless case, correlation-based methods cannot identify inherent all-pass factors, and cannot resolve the phase of the system.
- Non-Gaussian processes are not completely characterized by their second-order statistics; by using higher-order statistics, we are exploiting more of the information contained in the data.

A note of caution: for odd-ordered cumulants, $\gamma_{ku} \neq 0$ does not imply that the kth order polyspectrum can be used to reconstruct $H(f)$. For example, even if $\gamma_{3u} \neq 0$, $S_{3x}(f_1,f_2)$ in (1-20) may be identically equal to zero. This can happen when $H(f)$ is a relatively narrow-band bandpass signal; explicit conditions are given in [63]. An even more trivial example is $\gamma_{1u} = E\{u(n)\} \neq 0$, but $H(0) = 0$, in which case $\gamma_{1x} = 0$. For polyspectra of even order, say $2k$,

$$S_{2k}(f,-f,f,\ldots,-f,f) = \gamma_{2k,u}\left|S_{2x}(f)\right|^k / \gamma_{2,u}^k$$

cannot be identically zero if $\gamma_{2,u} \neq 0$ [63].

The transfer function $H(z) = B(z)/A(z)$ is said to have an inherent all-pass factor, if a root of $B(z)$ lies at $1/z_o$, where z_o is a root of $A(z)$. The power spectrum is blind to all-pass factors. If the data are noise free, and if the ARMA model does not have any inherent all-pass factors, then techniques based on the autocorrelation/power spectrum may be used to estimate the model parameters.

Once the ARMA parameters have been estimated, MATLAB's routine `freqz` can be used to estimate the transfer function and theoretical spectrum; Higher-Order Spectral Analysis Toolbox routines `bispect` and `trispect` can be used to compute the theoretical bispectrum and slices of the theoretical trispectrum corresponding to the ARMA model.

Synthetic ARMA processes can be generated via routine `armasyn`; routine `rpiid` can be used to generate i.i.d. sequences with various probability density functions (pdf's).

MA Models

Let us consider the pure MA case, that is, $p = 0$, in (1-31) through (1-37). It was established in [19] that the process $y(n)$ satisfies the set of equations,

$$C_{2y}(n) = \sum_{k=0}^{q} \varepsilon_3 b(k) C_{3y}(n-k, n-k) - \sum_{k=1}^{q} b^2(k) C_{2y}(n-k) \qquad \textbf{(1-38)}$$

$$C_{2y}(n) = \sum_{k=0}^{q} \varepsilon_4 b(k) C_{4y}(n-k, n-k, n-k) - \sum_{k=1}^{q} b^3(k) C_{2y}(n-k) \qquad \textbf{(1-39)}$$

where $n = -q, \ldots, 2q$; $\varepsilon_3 = \gamma_{3u} b(q)/\gamma_{2u}$, and $\varepsilon_4 = \gamma_{4u} b(q)/\gamma_{2u}$. Equations (1-38) and (1-39) can be readily verified by using the impulse response summation formulas in (1-16) through (1-18). These equations are sometimes referred to as the *GM equations* in the literature [19].

Equations (1-38) and (1-39) represent a set of $3q + 1$ linear equations in the $2q + 1$ unknowns, $\varepsilon_{m+1} b(k)$, $k = 0, \ldots, q$ and $b^m(k)$, $k = 1, \ldots, q$, where $m = 2$ or $m = 3$. In [19], MA parameters were estimated from (1-38) or (1-40) by simultaneously solving for the $\varepsilon_{m+1} b(k)$'s and $b^2(k)$'s or $b^3(k)$'s. A disadvantage of the method is that it is overparameterized, and does not take into account the relationship between the $\varepsilon_{m+1} b(k)$'s and the $b^2(k)$'s [or $b^3(k)$'s].

Additive white noise may be permitted, provided we eliminate the equations involving $C_{2y}(0)$; this eliminates $q + 1$ equations, leaving us only $2q$ equations in the $2q + 1$ unknowns. The Min-Norm solution is not useful in this case.

A modification developed by Tugnait [72] appends the following sets of equations to the preceding set of $2q$ equations:

$$\varepsilon_3 C_{2y}(n) - \sum_{k=1}^{q} b(k) C_{3y}(k-n, q) = C_{3y}(-n, q) \qquad \textbf{(1-40)}$$

$$\varepsilon_4 C_{2y}(n) - \sum_{k=1}^{q} b(k) C_{4y}(k-n, q, 0) = C_{4y}(-n, q, 0) \qquad \textbf{(1-41)}$$

where $n = -q, \ldots, q$, $n \neq 0$. Now, we have $4q$ equations in $2q + 1$ unknowns; hence, we can obtain the least-squares solution.

We solve either (1-38) and (1-40), or (1-39) and (1-41). Since both $b(k)$ and $b^2(k)$ or $b^3(k)$ are estimated, we need some method for combining the estimates.

Let $b_1(k)$ and $b_2(k)$ denote the estimates of $b(k)$ and $b^2(k)$. If all of the estimated $b^2(k)$'s are nonnegative, then the final MA parameter estimate is obtained as,

$$\hat{b}(k) = \text{sign}[b_1(k)] * \sqrt{0.5[b_1^2(k) + b_2(k)]};$$

otherwise, $\hat{b}(k) = b_1(k)$.

When fourth-order cumulants are used, the method estimates both $b(k)$ and $b^3(k)$. Let $b_1(k)$ and $b_3(k)$ denote the estimates of $b(k)$ and $b^3(k)$. If all the estimated $b_3(k)$'s have the same sign as the corresponding $b_1(k)$'s, then the final MA parameter estimate is obtained as,

$$\hat{b}(k) = \text{sign}[|b_1(k)|] * [|b_1(k)| + |b_3(k)|^{1/3}]/2;$$

otherwise, $\hat{b}(k) = b_1(k)$.

This algorithm is implemented in routine `maest`. If the observed process is $z(n) = y(n) + g(n)$, where $g(n)$ is additive noise, independent of $y(n)$, then $C_2z(n) = C_2y(n) + C_{2g}(n)$, $\forall n$. If $g(n)$ is white, $C_{2g}(n) = 0$, $n \neq 0$; hence, $C_2z(n) = C_2y(n)$, $n \neq 0$. In (1-40) and (1-41), we use the autocorrelation $C_2y(n)$, $n = \pm 1, \ldots, \pm q$, hence, the algorithm can handle additive white noise but not colored noise.

Examples

```
load ma1
bvec = maest (y,3,3,128);
```

The estimated parameters should be [1, 0.9714, 0.3814, –0.7759]. The true parameters are [1, 0.9, 0.385, –0.771], and the signal is contaminated with white Gaussian noise.

```
bvec = maest (y,3,4,256);
```

The estimated parameters should be [1, 0.9608, 0.4482, –0.7343].

AR Models

The cumulants of the noisy process satisfy the "normal" equations,

$$\sum_{k=0}^{p} a(k) C_{2y}(m-k) = 0 \qquad m > q \tag{1-42}$$

$$\sum_{k=0}^{p} a(k) C_{3y}(m-k, \rho) = 0 \quad , \quad m > q \tag{1-43}$$

$$\sum_{k=0}^{p} a(k) C_{4y}(m-k, \rho, \tau) = 0 \quad , \quad m > q \ , \tag{1-44}$$

which can be verified by expressing the cumulants in terms of the IR, $h(n)$, using (1-16) through (1-18), and by noting that $\sum_{k=0}^{p} a(k) h(n-k) = b(n)$.

The relationship between *linear prediction* (LP) and AR models is discussed in detail in the section titled "Polyspectra and Linear Processes." Here, we will just point out that the least squares solution to the LP problem is given by (1-42), with $m = 1, \ldots, p$; these equations are called the *normal* equations because the resulting prediction-error sequence is orthogonal to the data. However, when $q > 0$ or when additive noise is present in the data, the *normal equations* yield inconsistent estimates of the AR parameters. Equations (1-42) through (1-44) yield consistent estimates, and can also be derived by demanding that the prediction error sequence be orthogonal to an instrumental process derived from the data [59, 64]. We put *normal* in quotes to emphasize these differences.

Identifiability of the AR parameters is guaranteed by choosing $p = q - p, \ldots, q$, and $m = q + 1, \ldots, q + p$; we may use more slices (ρ) or more lags (m) [67]. In practice, we use sample estimates of the cumulants. This algorithm is implemented in routine `armarts`. The pure AR case corresponds to $q = 0$ and is implemented in routine `arrcest`. In both routines, you can use cumulant orders (2, 3, or 4). It is also possible to simultaneously solve for the *normal* equations based on cumulant orders 2 and 3 or 2 and 4.

Examples

Try the following

```
load ar1
ar(:,1)=arrcest (y,2,0,2,12,128);
ar(:,2)=arrcest (y,2,0,3,12,128);
ar(:,3)=arrcest (y,2,0,4,12,128);
ar(:,4)=arrcest (y,2,0,–3,12,128);
ar(:,5)=arrcest (y,2,0,–4,12,128);
disp(ar)
```

```
   1.0000    1.0000    1.0000    1.0000    1.0000
   -1.4636   -1.5559   -1.4963   -1.4912   -1.4755
   0.7664    0.8779    0.8917    0.7973    0.7927
```

The true parameters are [1, –1.5, 0.8]. The five columns correspond to AR estimates based on: (1) second-order, (2) third-order, (3) fourth-order, (4) combined second-order and third-order, and (5) combined second-order and fourth-order cumulants. Note that combined use of autocorrelation and cumulants may give better results when the signal-to-noise ratio (SNR) is high. In the case of low or moderate SNR, the correlation-based estimates will be biased; estimates based on third-order (fourth-order) cumulants will be unbiased if the additive noise is symmetric (Gaussian).

ARMA Models

As discussed in the previous section, we can determine the AR parameters easily. We will ignore the estimation errors, and assume that $\hat{a}(k) = a(k)$, $k = 1, \ldots, p$; this is justified if the data lengths are long enough to ensure good estimates of the cumulants. In practice, errors in estimating the $a(k)$'s will show up as an additive non-Gaussian noise term on the right-hand side of (1-46).

Consider the residual time series obtained via,

$$z(n) = \sum_{k=0}^{p} a(k)y(n-k) \qquad \textbf{(1-45)}$$

$$= \sum_{k=0}^{q} b(k)u(n-k) + \sum_{k=0}^{p} a(k)w(n-k)$$

$$= \sum_{k=0}^{q} b(k)u(n-k) + \sum_{k=0}^{p} w_1(n) \qquad \textbf{(1-46)}$$

Routine `armarts` uses the residual time series method to estimate the ARMA parameters — it estimates the AR parameters first (via the "normal" equations); it then computes the AR-compensated time series via (1-45), and finally estimates the MA parameters via routine `maest` (see (1-46)). Since $w(n)$ was assumed to be Gaussian, $w_1(n)$ is also Gaussian; if $w(n)$ is white, $w_1(n)$ is MA(p) noise. Routine `maest` assumes that the additive noise is white, hence the results of `armarts` are meaningful only at high SNR.

An alternative solution based on is implemented in routine `armaqs`, which is a q-slice method. The AR parameters are estimated via the normal equations as before. The impulse response is then estimated via,

$$h(n) = \frac{\sum_{k=0}^{p} a(k)C_{3y}(q-k, n)}{\sum_{k=0}^{p} a(k)C_{3y}(q-k, 0)}, \quad n = 1, \ldots, q \qquad \textbf{(1-47)}$$

or via,

$$h(n) = \frac{\sum_{k=0}^{p} a(k)C_{4y}(q-k, n, 0)}{\sum_{k=0}^{p} a(k)C_{4y}(q-k, 0, 0)}, \quad n = 1, \ldots, q \qquad \textbf{(1-48)}$$

These equations can be readily verified by using (1-16) through (1-18) and (1-35). The MA parameters are then obtained via (1-35), which is repeated here:

$$b(n) = \sum_{k=0}^{p} a(k)h(n-k), \quad n = 1, \ldots, q. \qquad \textbf{(1-49)}$$

An interesting point is that we can simultaneously solve for the AR and IR parameters. This algorithm is implemented in routine armaqs.

Weighted versions of (1-42) through (1-44), and nonlinear *cumulant-matching* algorithms for the simultaneous estimation of AR and MA parameters are discussed in [16] and [71].

Examples

```
load arma1
[avec,bvec]=armaqs(y,2,1,3,10,128);
```

Here we used third-order cumulants and the q-slice algorithm to estimate the parameters of a non-Gaussian ARMA process. The estimated parameters should be avec = [1, –0.8057, 0.6910], and bvec = [1, –1.9116]. The true AR and MA parameters were [1, –0.8, 0.65] and [1, –2], respectively.

```
[avec, bvec] = armarts(y,2,1,3,12,128);
```

Here we used third-order cumulants and the residual time-series algorithm to estimate the parameters of a non-Gaussian ARMA process. The estimated parameters should be avec = [1, –0.7540, 0.6465], and bvec = [1, –1.5665]. The true AR and MA parameters were [1, –0.8, 0.65] and [1, –2], respectively.

AR Order Determination

Let $\bar{p}$ and $\bar{q}$ denote the maximum expected values of the AR and MA orders. Let

$$\mathbf{c}_{2y}(m) := C_{2y}(m) \tag{1-50}$$

$$\mathbf{c}_{3y}(m) := [C_{3y}(m-\bar{p}), \ldots, C_{3y}(m,\bar{q}]^T \tag{1-51}$$

$$\mathbf{c}_{4y}(m) := [C_{4y}(m-\bar{p},0), \ldots, C_{4(3)y}(m,\bar{q},0]^T \tag{1-52}$$

Then, the singular values of the matrix,

where $k = 2$, $k = 3$, or $k = 4$ are computed.

Let s(m) denote the singular values. If the cumulant estimates are perfect, we expect that exactly p of the singular values will be nonzero; with sample estimates, we expect p dominant singular values. The AR order p is

$$\mathbf{C}_k = \begin{bmatrix} \mathbf{c}_{ky}(\bar{q}+1) & \dots & \mathbf{c}_{ky}(\bar{q}+\bar{p}) \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{ky}(\bar{q}+\bar{p}+1) & \dots & \mathbf{c}_{ky}(\bar{q}+2\bar{p}-1) \end{bmatrix}$$

then given by the value of n, which maximizes $s(n) - s(n+1)$, that is, it corresponds to the index at which the singular values show the maximum drop. This is a nonstatistical test, and is based on the fact that for an ARMA(p,q) model, only p of the singular values should be nonzero. This algorithm is implemented in routine `arorder`.

Examples

```
load arma1
p=arorder(y,3);
```

You should see the display in Figure 1-6. The estimated AR order is 2.

The time-series y corresponds to a non-Gaussian ARMA(2,1) process, contaminated by AWGN, with SNR of 20 dB. The order determination is based on third-order cumulants.

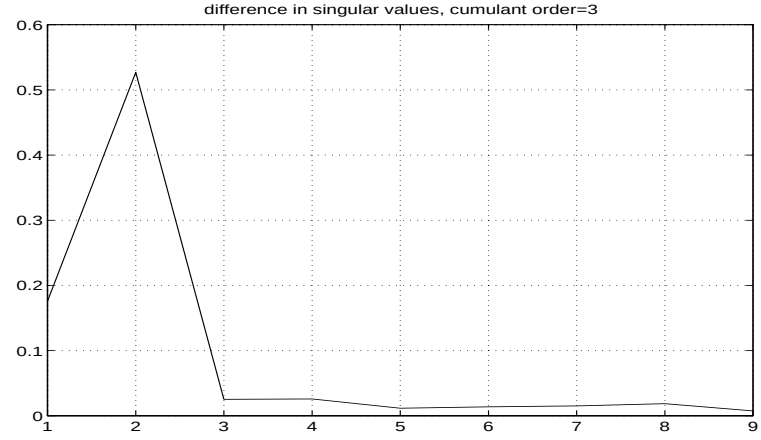


Figure 1-6 Estimate of AR Order Using SVD Method (arorder)

MA Order Determination

The basic idea is that for an MA(q) process, the true values of the cumulant $C_{3y}(m,0)$ will be identically zero, if $m > q$. When the true cumulants are replaced by sample estimates, the estimated values of $C_{3y}(q+i,0)$, $i > 0$ will not be identically zero; a statistical test is used to determine whether the estimated values are close to zero. This test is based on estimating the theoretical variance of the sample estimates of $C_{3y}(m,0)$.

Sample estimates, $\hat{C}_{3y}(m,0)$, and their variances are estimated for m ranging from *qmin* to *qmax*, which reflect our *a priori* knowledge of the bounds on the true order q.

For an MA(q) process, the asymptotic variance of the sample estimate of $C_{3y}(q+1,0)$ can be estimated via [20]

$$\hat{\sigma}^2(q+1) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=-q}^{2q+1}\left(1-\frac{|j|}{N}\right)[y^2(i)y(i+q+1)-\hat{c}_{3y}(q+1,0)] \times [y^2(i+j)y(i+j+q+1)-\hat{c}_{3y}(q+1,0)]$$

where N is the length of the time series.

The sample estimates are asymptotically Gaussian and unbiased; hence, the threshold t_c in

$$\Pr\{|\hat{c}_{3y}(m+1,0)| \le t_c(m+1)\} = 1 - pfa$$

is given by

$$t_c(m+1) = erfinv[(1-pfa)\sqrt{2\sigma^2(m+1)}],$$

where `erfinv` is the MATLAB inverse error function; in practice, we use the sample estimate σ^2. Let m_o denote the largest value of m in the range *qmin* to *qmax* for which $|c_{3y}(m+1,0)| > t_c(m+1)$ (so that the hypothesis of MA(m_o) model fails); then, the estimated order is $q = m_o + 1$; if such an m does not exist, the MA order is declared to be $qmax + 1$. This algorithm is implemented in routine `maorder`.

This is a statistical test, and *pfa* specifies the fraction of the time that the test results will be wrong. In other words, in a Monte Carlo simulation of 1000 trials, one should expect the test results to be wrong $1000 * pfa$ times.

Examples

```
load ma1;
q=maorder(y,0,6);
```

The following table will be displayed:

q	var(cqk)	cqk	thres	result
0	7.17383e-003	7.31541e-001	5.24957e-001	0
1	6.60864e-002	-1.37221e-001	1.59332e-001	1
2	3.76124e-002	-4.11813e-001	3.80114e-001	0
3	7.04832e-003	5.06864e-003	1.64547e-001	1
4	3.77757e-003	-3.31784e-002	1.20463e-001	1
5	1.31938e-003	-6.87545e-002	7.11923e-002	1
6	4.47359e-003	1.43717e-003	1.31092e-001	1
Estimated MA order is 3				

The columns in the table correspond to the estimated variance of $c_{3y}(q,0)$, the estimated value of $c_{3y}(q,0)$, the corresponding threshold, and whether or not the absolute value of the estimated $c_{3y}(q,0)$ exceeded the threshold.

The time-series y corresponds to an MA(3) process, contaminated by AWGN, with SNR of 20 dB.

Linear Processes: Impulse Response Estimation

The basic model here is,

$$x(n) = \sum_{k=-\infty}^{\infty} h(k)u(n-k) \tag{1-53}$$

$$y(n) = x(n) + w(n) \tag{1-54}$$

where additive noise $w(n)$ is assumed to be symmetric distributed (not necessarily Gaussian). Process $u(n)$ is i.i.d., non-Gaussian, independent of the noise, and satisfies $0 < |C_{3u}(0,0)| < \infty$.

We will discuss two algorithms, both based on the notion of the logarithm of the bispectrum.

The Polycepstral Methods

It is assumed that $H(z)$ has no zeros on the unit circle. In this case, the cepstrum of $H(z)$ is well defined and is given by [42]

$$\hat{h}(k) = \int df \exp(j2\pi f k) \ln H(f) \tag{1-55}$$

The bicepstrum is also well defined [43] and is given by,

$$\hat{b}(m, n) = \int \int df_1 df_2 e^{j2\pi f_1 m} e^{j2\pi f_2 n} \ln S_{3x}(f_1, f_2) \tag{1-56}$$

$$= \hat{h}(m)\delta(m) + \hat{h}(n)\delta(n) + \hat{h}(-m)\delta(m-n) + \ln \Upsilon_{3u} \delta(m)\delta(n) \tag{1-57}$$

The complex cepstrum, $\hat{h}(k)$, is the inverse FT (IFT) of the log of the FT of $h(n)$; hence, $h(n)$ is the IFT of the exponential of the FT of the complex cepstrum; other techniques to obtain $h(n)$ from $\hat{h}(k)$ are described in [42]. Notice that for a linear process, the bicepstrum is nonzero only along the axes, $m = 0$, $n = 0$, and the diagonal line, $m = n$.

Direct implementation of (1-56) demands 2-D phase unwrapping. Pan and Nikias [43] have developed an alternative method, based on the relationship,

$$\sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} m \hat{b}(m, n) C_{3y}(k-m, l-n) = k C_{3y}(k, l) \tag{1-58}$$

which, taking into account (1-57), reduces to

$$\sum_{m=-\infty}^{\infty} m \hat{h}(m)[C_{3y}(k-m, n) + C_{3y}(k+m, l+m)] = k C_{3y}(k, l)$$

The complex cepstrum is known to be exponentially bounded; hence, one may replace the infinite summation over m by a finite summation, $m = -q$ to $m = +p$,

where p and q may be based on some *a priori* knowledge; note that the p and q have nothing to do with the orders of an ARMA(p,q) process. Now we have a set of linear equations for estimating a finite set of $p + q$ parameters.

$$\sum_{m=-q}^{p} m\hat{h}(m)[C_{3y}(k-m,n) + C_{3y}(k+m,l+m)] = kC_{3y}(k,l) \qquad \textbf{(1-59)}$$

Note that the above equation does not involve $\hat{h}(0)$; the cepstrum at the origin is related to the overall gain of the system. Because of the inherent scalar ambiguity in estimating $H(z)$ from power spectra or cumulant spectra, we let $\hat{h}(0) = 1$. In practice sample estimates of the third-order cumulants are used; this algorithm is implemented in routine `biceps`, where $|k| \le \max(p,q)$ and $|l| \le \max(p,q)/2$, which are the recommended ranges in [43].

Examples

```
load ma1
[hest, ceps]=biceps(y,8,8,128);
```

You should see the display in Figure 1-7. Note that the complex cepstrum decays rapidly to zero. The true MA parameters are [1, 0.9, 0.385, –0.771].

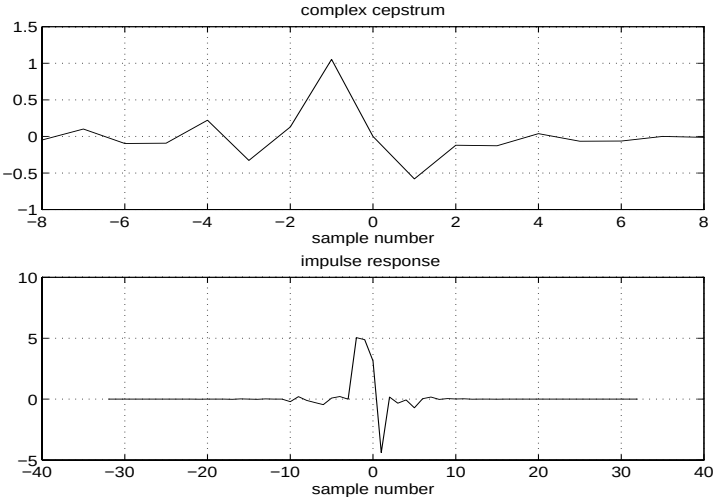


Figure 1-7 Cepstrum and IR Estimated by Biceps

An alternative approach is based on the FFT. We may rewrite (1-58) as [43]

$$m\hat{b}(m,n) = IFT\left(\frac{FT(kC_{3y}(k,l))}{FT(C_{3y}(k,l))}\right) \tag{1-60}$$

where IFT denotes 2-D inverse Fourier transform. Then we make use of (1-57) to obtain the $\hat{h}(k)$'s from $m\hat{b}(m,n)$. The FFT approach is useful if the cepstrum $\hat{h}(k)$ has long support. This algorithm is implemented in routine `bicepsf`.

More generally, the polycepstrum is defined as the inverse FT of the log of the corresponding polyspectrum (assuming that it exists),

$$\hat{b}_{kx}(m_1,\ldots,m_{k-1}) = \int df_1 \ldots \int df_{k-1} e^{j2\pi f_1 m_1 \ldots ej2\pi^{f_{k-1}} m_{k=1}} \ln S_{kx}(f_1,\ldots,f_{k-1})$$

For a linear process, the polycepstrum is nonzero only along the $k-1$ axes, and the main diagonal,

$$\hat{b}_{kx}(m_1,\ldots,m_{k-1}) = \sum_{l=1}^{k-1} \hat{h}(m_l) \prod_{n\neq l;n=1}^{k-1} \delta(m_n) + \hat{h}(-m_l) \prod_{l=2}^{k-1} \delta(m_1 - m_l) + \ln\gamma_{ku} \prod_{l=2}^{k-1} \delta(m_l)$$

Tekalp and Erdem [69] showed a process, whose polycepstrum exists, is linear if and only if its polycepstrum has the above k-line region of support. Based on this, they have also proposed measures of linearity.

Examples

```
load ma1
[hest, ceps]=bicepsf(y, 8);
```

You should see the display in Figure 1-8. Note that the complex cepstrum decays rapidly to zero. The true MA parameters are [1, 0.9, 0.385, –0.771]; note the scale ambiguity (including sign) in the estimated impulse response.

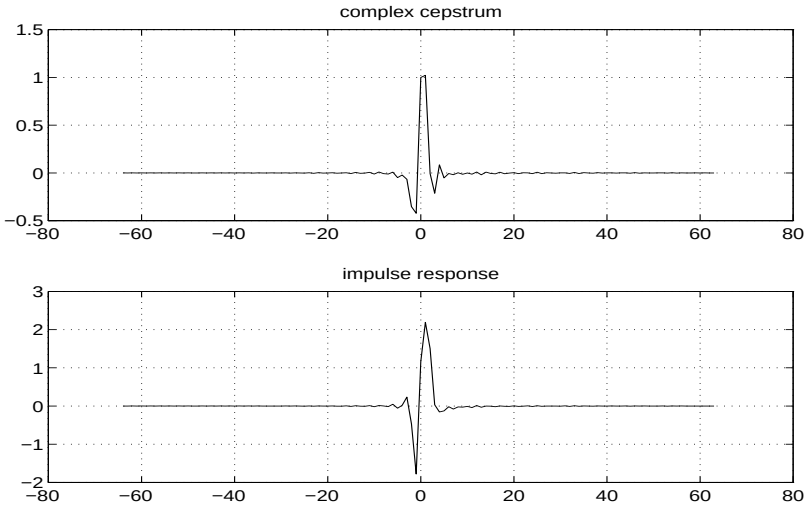


Figure 1-8 Cepstrum and IR Estimated by bicepsf

The Matsuoka-Ulrych Algorithm

We can write the bispectrum in terms of its magnitude and phase as,

$$S_{3x}(f_1,f_2) = M(f_1,f_2)\exp(j\Phi(f_1,f_2)).$$

Let $H(f) = |H(f)|\exp(j\theta(f))$ so that

$$\ln M(f_1, f_2) = \ln\left|\gamma_{3u}\right| + \ln\left|H(f_1)\right| + \ln\left|H(f_2)\right| + \ln\left|H(f_1 + f_2)\right| \quad \textbf{(1-61)}$$

$$\Phi(f_1, f_2) = \theta(f_1) + \theta(f_2) - \theta(f_1 + f_2) \mod 2\pi \quad \textbf{(1-62)}$$

where f_1,f_2 take on discretized values on a grid. The assumption that $|\gamma_{3u}| = 1$ will only introduce an overall scalar ambiguity; hence, (1-61) represents a set of linear equations in the log-magnitude $\ln |H(f)|$. It is shown in [36] that a full rank set of linear equations can be obtained by appropriate choice of f_1 and f_2. The phase relationship in (1-62) holds only modulo 2π. If the unwrapped 2-D phase were available, then (1-62) yields a linear set of equations in the phase of the transfer function, $\theta(f)$. This algorithm is implemented in routine `matul`, where we also incorporate the phase-unwrapping algorithms of [54]. Because of the phase ambiguity, this routine is not recommended for routine use.

Examples

```
cmat=cumtrue([1 –3.5 1.5],[1],3,5);
bsp=fft2(flipud(cmat),64,64);
hest=matul(bsp);
```

You should see the display in Figure 1-9. Here we computed the true bispectrum of a MA(2) process, by evaluating the FT of its true cumulant sequence. We then used the Matsuoka-Ulrych algorithm to estimate the impulse response. Note the scale ambiguity (including sign).

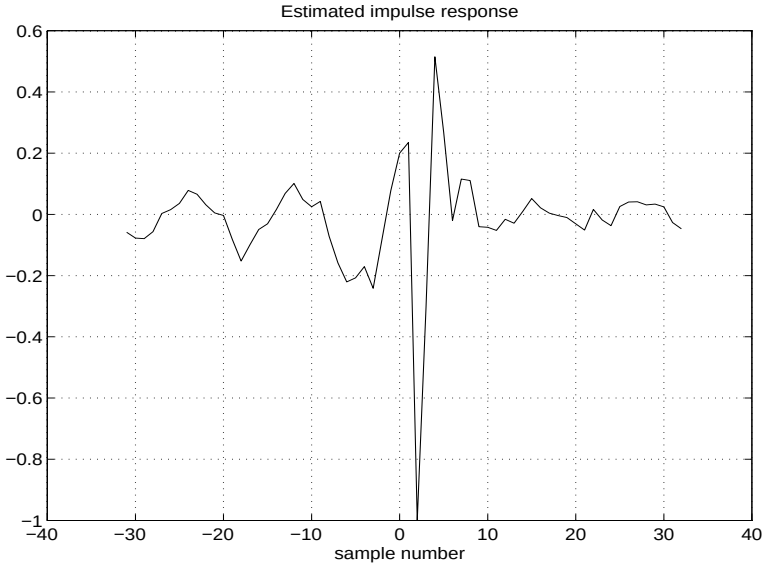


Figure 1-9 IR Estimated via the Matsuoka-Ulrych Algorithm (matul)

Linear Processes: Theoretical Cumulants and Polyspectra

If $x(n)$ is an ARMA process, that is,

$$x(n) = \sum_{k=1}^{p} x(n-k) + \sum_{k=0}^{q} b(k)u(n-k),$$

then, its impulse response can be calculated via $h(n) = -\sum_{k=1}^{p} h(n-k) + b(n)$, where $h(n) = 0$ for $n < 0$, $h(0) = 1$, and $b(n) = 0$, if $n \notin [0, q]$.

For a linear process $x(n)$ ($x(n) = \sum_k h(k)u(n-k)$, where $u(n)$ is i.i.d.) cumulants and polyspectra of orders 2,3,4 are given by (1-16) through (1-18) and (1-19) through (1-21).

We can estimate the true cumulants using routine `cumest`, the theoretical bispectrum via routine `bispect`, and slices of the theoretical trispectrum via routine `trispect`.

Examples

First, we will compute the theoretical third-order cumulants of an ARMA(2,1) process, with AR parameters, [1, –1.5, 0.8], and MA parameters, [1, –2]; we will then display the estimates using MATLAB's functions `mesh` or `contour`:

```
cmat=cumtrue([1,-2],[1,-1.5,0.8],3,25);
clf, subplot(121)
mesh(-25:25,-25:25,cmat), grid on
subplot(122), contour(-25:25,-25:25,cmat,8,),grid on
```

You should see the display in Figure 1-10.

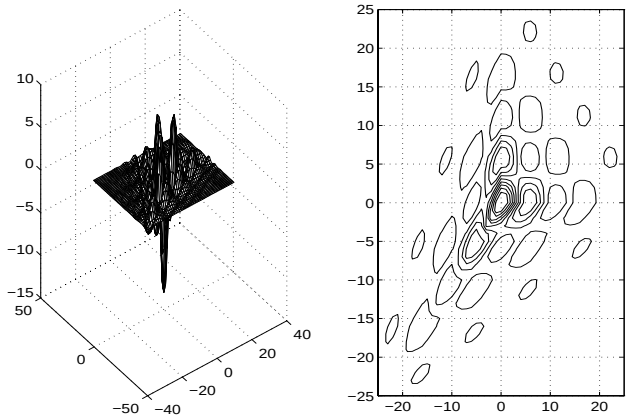


Figure 1-10 True Third-Order Cumulants of an ARMA(2,1) Process (cumtrue)

Let us now compute the theoretical bispectrum of another ARMA(2,1) process.

```
ma=[1 –2]; ar=[1 –0.8 0.65];
bisp=bispect (ma,ar,128);
```

You should see the display on Figure 1-11. The 12 dotted lines emanating from the origin divide the bifrequency domain of the bispectrum into 12 regions of symmetry.

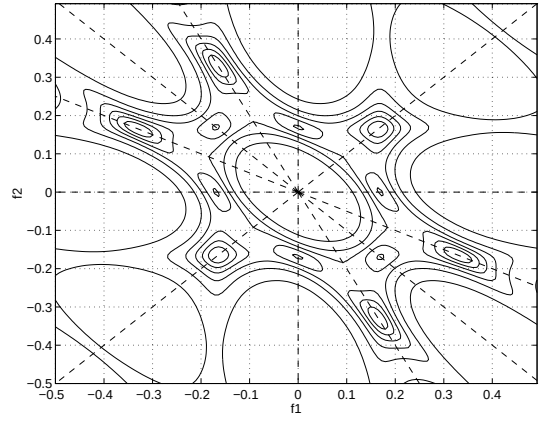


Figure 1-11 Bispectrum of an ARMA(2,1) Process (bispect)

We will compute several slices of the trispectrum and then display them as a *movie*:

```
clf,clear
ma=[1 –2]; ar=[1 –0.8 0.65]; nfft=64;
n=10; M=moviein(2*n+1);
for k=–n:n
    trispect(ma,ar,nfft,k/(2*n));
    M(:,k+n+1)=getframe;
end
movie (M)
clear M
```

We computed several slices of the theoretical trispectrum of an ARMA process; the slices correspond to $f_3 = –0.5, –0.45, \ldots, 0.45, 0.5$. Then we used MATLAB's `movie` command to display them; `moviein` and `getframe` are also MATLAB commands. It is instructive to change the MA parameters to [1 –1], and to compare the
$f_3 = 0$ slice with the theoretical bispectrum created by `bispect`.

NOTE: if you encounter "`OUT OF MEMORY`" problems in running the example, clear the workspace and then try again. If the problem persists, decrease `n` to, say, 5; or install more memory on your machine.

Summary

We have discussed various algorithms for estimating bispectra and trispectra, as well as algorithms for the blind system identification problem.

Sample estimates of cross-cumulants of orders 2, 3 and 4 may be obtained via `cum2x`, `cum3x`, and `cum4x`; autocumulants can be estimated via `cumest`. Nonparametric estimates of the bispectrum can be obtained via routines `bispeci` and `bispecd`, which implement the indirect and direct estimators. The direct estimate of the cross-bispectrum can be obtained via routine `bispecdx`. Auto- and cross-bicoherences can be estimated via `bicoher` and `bicoherx`. The theoretical bispectrum of a linear process can be computed via `bispect`; `trispect` computes slices of the theoretical trispectrum of a linear process. M-file `glstat` implements tests for non-Gaussianity (actually nonzero bispectrum) and for linearity.

Three routines are available for nonparametric estimation of the impulse response of a linear non-Gaussian process: `matul` implements the Matsuoka-Ulrych method; given the bispectrum of the data, it estimates the amplitude and phase of the transfer function separately; `biceps` uses the bicepstral method; it uses sample estimates of the third-order cumulant; `bicepsf` is a frequency-domain implementation of `biceps` and is useful if the cepstra do not decay fast enough.

Several algorithms are available for estimating the (ARMA) parameters of non-Gaussian linear processes. The ARMA orders can be estimated via `arorder` and `maorder`. Routine `arrcest` can be used to estimate the AR parameters of AR or ARMA processes, based on second-, third-, and fourth-order cumulants. The estimates, based on cumulants of different orders, may not be the same, if the process is nonlinear or has inherent all-pass factors, or is noisy. The parameters of an MA process, contaminated by white noise, may be estimated via routine `maest`.

ARMA parameters can be estimated via the residual time-series algorithm in `armarts`; the MA estimation part of this three-step algorithm works well only under good SNR conditions. Routine `armaqs` implements the q-slice algorithm; it estimates both AR and MA parameters.

In this section, we have given a quick overview of the area of higher-order statistics; for more extended tutorial expositions, see [41, 37, 38].

Linear Prediction Models

The AR model is also obtained if one considers the problem of linear prediction (LP). Depending upon whether one chooses forward-, backward- or forward-backward prediction criteria, different solutions are obtained.

The Levinson algorithm enables efficient estimation of the parameters of an AR process from its autocorrelation (AC) sequence. In the deterministic context, AR modeling leads to the forward backwards least squares (FBLS) solution. The RLS algorithm is a time-recursive solution to the LP problem, whereas the lattice algorithm provides a time- and order-recursive solution.

Levinson Recursion

Consider the linear prediction problem for a stationary process, $x(n)$. In the forward-prediction problem, we want to choose $\{a_p(k)\}_{k=1}^{p}$ to minimize the forward prediction error variance, P_p^f,

$$P_p^f = E\left\{\left|e_p^f(n)\right|^2\right\} \tag{1-63}$$

$$e_p^f(n) = \sum_{k=0}^{p} a_p^*(k)x(n-k), \quad a_p(0) = 1 \tag{1-64}$$

where the subscript p denotes the order of the predictor. Similarly, in the backward-prediction problem, we want to choose $\{c_p(k)\}_{k=1}^{p}$ to minimize the backward prediction error variance, P_p^b,

$$P_p^b = E\left\{\left|e_p^b(n)\right|^2\right\} \tag{1-65}$$

$$e_p^b(n) = \sum_{k=0}^{p} c_p^*(k)x(n-k), \quad c_p(p) = 1 \tag{1-66}$$

This leads to the normal equations

$$\Re \mathbf{a}_p = \begin{bmatrix} P_p^f \\ 0 \end{bmatrix} \tag{1-67}$$

$$\Re \mathbf{c}_p = \begin{bmatrix} 0 \\ P_p^b \end{bmatrix}, \tag{1-68}$$

where $\Re$ is the $(p + 1) \times (p + 1)$ autocorrelation matrix, with (m,n) entry, $R_{xx}(n - m)$. It follows immediately that

$$c(k) = a^*(p - k), \quad k = 0, \ldots, p; \quad P_p^f = P_p^b. \tag{1-69}$$

The Levinson-Durbin recursion implemented in routine `trench` provides an efficient order-recursive solution of the normal equations: it recursively solves (1-67), for $p = 1,2,\ldots$.

The recursion is given by: for $m = 1, \ldots, p$, [32]

$$\Delta_{m-1} = \sum_{k=0}^{m-1} a_{m-1}(k) R_{xx}(k - m) \tag{1-70}$$

$$\Gamma_m = -\frac{\Delta_{m-1}}{P_{m-1}} \tag{1-71}$$

$$P_m = P_{m-1}(1 - |\Gamma_m|^2) \tag{1-72}$$

$$a_m(k) = a_{m-1}(k) + \Gamma_m, a^*_{m-1}(m - k), \quad k = 0, \ldots, m \tag{1-73}$$

with initial conditions $P_0 = R_{xx}(0)$, $\Delta_0 = R_{xx}(-1)$, and $a_0(0) = 1$. In practice, we replace $R_{xx}(k)$ by its (biased) sample estimate. The resulting solution is

guaranteed to be stable if biased estimates are used. Necessary and sufficient conditions for the stability of the estimate are $|\Gamma_m| < 1 \ \forall_m$.

The recursion involves computation of the Γ_m's, which are also called the reflection coefficients; a necessary and sufficient condition for stability is $|\Gamma_m| < 1$.

Trench Recursion

If the matrix $\Re$ is Toeplitz but not Hermitian symmetric, then (1-69) does not hold; this, for example, is the case when the so-called higher-order Yule Walker equations,

$$\sum_{k=0}^{p} a_p(k) R_{xx}(m-k) = 0, \quad m = q+1, \ldots, q+p \tag{1-74}$$

are used, or when the *cumulant-based normal equations* are used, for example,

$$\sum_{k=0}^{p} a_p(k) C_{3x}(m-k, \rho) = 0, \quad m = q+1, \ldots, q+p. \tag{1-75}$$

In both cases the resulting matrix is Toeplitz but not symmetric. An efficient order-recursive solution to the system of equations is given by the Trench recursion [74].

Let $\Re$ denote the nonsymmetric Toeplitz matrix whose (i,j) entry is $\rho(i-j)$. The recursion is given by

$$\Delta_{m-1}^{f} = \sum_{k=0}^{m-1} a_{m-1}(k) \rho(m-k) \tag{1-76}$$

$$\Delta_{m-1}^{b} = \sum_{k=0}^{m-1} c_{m-1}(k) \rho(-1-k) \tag{1-77}$$

$$\Gamma_m^f = -\frac{\Delta_{m-1}^f}{P_{m-1}} \tag{1-78}$$

$$\Gamma_m^b = -\frac{\Delta_{m-1}^b}{P_{m-1}} \tag{1-79}$$

$$P_m = P_{m-1}(1 - \Gamma_m^f * \Gamma_m^b) \tag{1-80}$$

$$a_m(m) = \Gamma_m^f \tag{1-81}$$

$$c_0(m) = \Gamma_m^b \tag{1-82}$$

$$a_m(k) = a_{m-1}(k) + \Gamma_m^f c_{m-1}(k-1), \quad k = 1, \ldots, m-1 \tag{1-83}$$

$$c_m(k) = c_{m-1}(k-1) + \Gamma_m^b a_{m-1}(k), \quad k = 1, \ldots, m-1 \tag{1-84}$$

with initial conditions $P_0 = \rho(0)$, $a_0(0) = 1$, and $c_0(0) = 1$.

The inversion algorithm requires that the matrix $\Re$ be strongly nonsingular, that is, R, as well as all of its principal minors, are nonsingular. The condition of nonsingularity is weaker than the usual assumption of positive-definiteness.

In the symmetric positive-definite case, the Γ's are bounded by unity, and the P_m's are nonincreasing, and $c_m(k) = a_m(m-k)$, $k = 0, \ldots, m$. In the non-Hermitian case, the Γ's are not bounded, and the P_m's may increase or decrease.

This algorithm is implemented in `trench`.

Examples

Let us compute the autocorrelation sequence for an AR(4) model, with AR parameters [1,–0.3,–0.1,–0.39,0.72] , and then apply the Trench recursion.

```
ar=[1,-0.3,-0.1,-0.39,0.72]; ma=1; nlags=6;
rvec=cumtrue (ma,ar,2,nlags);
[a2,c2,p2,gf2,gb2]=trench( rvec(7:13), rvec(7:-1:1));
```

The AR matrix, a2, should be

1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-0.0764	-0.0700	-0.0399	-0.0300	-0.3000	-0.3000
0	-0.0834	-0.0581	-0.1000	-0.1000	-0.1000
0	0	-0.3613	-0.3900	-0.3900	-0.3900
0	0	0	0.7200	0.7200	0.7200
0	0	0	0	0.0000	0.0000
0	0	0	0	0	0.0000

Note that the AR vectors for orders greater than three are all essentially the same. The prediction error variance, p2, should be

```
[2.4049, 2.3881, 2.0764, 1.0000, 1.0000, 1.0000]'
```

The forward reflection coefficients, gf2, should be

```
[-0.0764, -0.0834, -0.3613, 0.7200, 0.0000, 0.0000]'
```

Since the Toeplitz matrix in this example is symmetric, gb2=gb1, and c2=flipud(a2). Since the autocorrelation sequence corresponds to an AR(4) model, the reflection coefficients are zero for $m > 4$, and the pfe's remain constant for $m \geq 4$.

Now let us repeat this example using third-order cumulants:

```
rvec=cumtrue (ma, ar, 3,          % third-order cumulants
nlags);
rvec=rvec(:,nlags+1);             % the C(m,0) slice
c=rvec(nlags+1:2*nlags);          % positive lags
r=rvec(nlags+1:-1:1);             % negative lags
[a4, c4, p4, gf4, gb4]=trench(c,r);
```

The forward AR matrix, a4, should be

1.0000	1.0000	1.0000	1.0000	1.0000
-0.2278	-0.2050	-0.1834	-0.3000	-0.3000
0	-0.1163	-0.0673	-0.1000	-0.1000
0	0	-0.2757	-0.3900	-0.3900
0	0	0	0.7200	0.7200
0	0	0	0	0.0000

The backward AR coefficient matrix, c4, should be

0	0	0	0	-0.0045
0	0	0	-0.2612	-0.2599
0	0	-0.1620	-0.1141	-0.1137
0	-0.0786	-0.0454	-0.0278	-0.0261
-0.1956	-0.1776	-0.1588	-0.0868	-0.0900
1.0000	1.0000	1.0000	1.0000	1.0000

and the *prediction-error variances*, p4, should be

```
[0.8892, 0.8810, 0.8417, 1.0000, 1.0000]'
```

The forward reflection coefficients, gf4, are,

```
[-0.2278, -0.1163, -0.2757, 0.7200, 0.0000]'
```

and the backward reflection coefficients, gb4, are

```
[-0.1956, -0.0786, -0.1620, -0.2612, -0.0045]'
```

Since the Toeplitz matrix is not symmetric, gf4≠gb4, and a≠4flipud(c4). Note also that the prediction error variances may increase with the prediction order. (One can find the AR parameters by minimizing the variance of the AR-compensated process, but not by minimizing the skewness.) Although the correlation and cumulant sequences are derived from the same AR model, the lower-order AR model fits are quite different.

Deterministic Formulation of FBLS

In the deterministic formulation of the linear prediction problem, rather than minimize the variance of the prediction errors, we minimize the sum of squared errors, that is,

$$P_p^b = \sum_{n=p+1}^{N} \left| e_p^b(n) \right|^2 \tag{1-85}$$

$$P_f^b = \sum_{n=p+1}^{N} \left| e_p^b(n) \right|^2 \tag{1-86}$$

This leads to the *deterministic normal equation*,

$$\Phi \mathbf{a}_p = \begin{bmatrix} P_p^f \\ 0 \end{bmatrix} \tag{1-87}$$

$$\Phi \mathbf{c}_p = \begin{bmatrix} 0 \\ P_p^b \end{bmatrix} \tag{1-88}$$

where the deterministic correlation matrix Φ is given by

$$\Phi = \sum_{n=p+1}^{N} \mathbf{x}(n)\mathbf{x}^H(n) \tag{1-89}$$

$$\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-p)]^T \tag{1-90}$$

Matrix Φ is Hermitian, but not Toeplitz, and the relationship in (1-69) is not satisfied (for finite N).

In the forward-backward least squares (FBLS) problem, we minimize $P_p = P_p^f + P_f^b$ assuming $c_p(k) = a_p^*(p-k)$. This leads to the deterministic normal equations,

$$\tilde{\Phi}\mathbf{a}_p = \begin{bmatrix} P_p \\ 0 \end{bmatrix} \tag{1-91}$$

where $\tilde{\Phi}$ is given by

$$\tilde{\Phi} = \sum_{n=p+1}^{N} \mathbf{x}(n)\mathbf{x}^H(n) + \tilde{\mathbf{x}}(n)\tilde{\mathbf{x}}^H(n) \tag{1-92}$$

where

$$\tilde{\mathbf{x}}(n) = [x(n-p), \ldots, x(n)]^H \tag{1-93}$$

If $\tilde{\Phi}$ is nonsingular, we can solve (1-91) directly; if it is singular, then, the Min-Norm solution is usually used.

The solution $\mathbf{a}_p$ may be interpreted as providing an AR(p) model for the data; hence, we can compute the spectrum of the process $x(n)$ in terms of the estimated AR parameters. Additional details may be found in [34, 35].

Adaptive Linear Prediction

The Burg algorithm provides a recursive solution to minimizing the mean-square error over a time interval of length n, and is appropriate when the data may be considered to be stationary over that time interval. Often, the data are nonstationary; in this case, we can estimate power spectra over a sliding window of fixed length; however, as we have seen before, the resulting estimate will have poor resolution and high variance. An alternative is to fit a parametric model, and to allow the model parameters to vary with time. For example, we may fit a time-varying AR model, that is, we perform time-varying linear prediction.

The deterministic normal equations at time n are given by (see (1-87) and (1-88))

$$\Phi(n)\mathbf{a}_p(n) = \begin{bmatrix} P_p^f(n) \\ 0 \end{bmatrix} \tag{1-94}$$

$$\Phi(n)\mathbf{c}_p(n) = \begin{bmatrix} 0 \\ P_p^b(n) \end{bmatrix} \tag{1-95}$$

where the deterministic correlation matrix $\Phi(n)$ is given by,

$$\Phi(n) = \sum_{k=p+1}^{n} \mathbf{x}(k)\mathbf{x}^H(k)\lambda^{n-k} \tag{1-96}$$

where λ, $(0 < \lambda < 1)$, is the forgetting factor, which is introduced so that the algorithm will be sensitive to the nonstationary environment.

We could solve (1-94) or (1-95) at each n, but the computational load is very high.

Let the order p be fixed, and let $\mathbf{a}_p(n) = [1, -\mathbf{w}(n)]^T$. Then, a time-recursive solution is given by,

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n)}{1 + \lambda^{-1}\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)} \tag{1-97}$$

$$\alpha(n) = x(n+1) - \mathbf{w}^H(n-1)\mathbf{x}(n) \tag{1-98}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\alpha^*(n) \tag{1-99}$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{x}^H(n)\mathbf{P}(n-1) \tag{1-100}$$

with initial values, $\mathbf{w}(0) = 0$ and $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$, where δ is a small positive constant.

$\mathbf{k}(n)$ is the *gain vector*, $\alpha(n)$ is the *a priori prediction error*, and $\mathbf{P}(n) = \Phi^{-1}(n)$. This algorithm is known as the Recursive Least Squares Algorithm (RLS), and is a special case of the Recursive Instrumental Variable Algorithm (RIV), which is discussed next.

RIV Algorithm: Transversal Form

Consider the noisy AR process,

$$y(n) = x(n) + w(n); \qquad x(n) = -\sum_{k=1}^{p} a(k)u(n-k)$$

where $u(n)$ is assumed to be zero mean i.i.d., and independent of additive noise $w(n)$, which may be colored. The "normal" equations, with $m = 1, \ldots, p$,

$$\sum_{k=0}^{p} a(k)E\{y(n)z(n-m+k)\} = 0, \qquad m > 0 \tag{1-101}$$

where $z(n) = y(n)$, lead to inconsistent estimates because of the noise $w(n)$. If $w(n)$ can be modeled as MA noise, that is,

$$w(n) = \sum_{k=0}^{q_w} h_w(n-k)g(n),$$

where $g(n)$ is i.i.d., then, consistent estimates can be obtained by using $z(n) = y(n-d)$, where $d > p + q_w$. Process $z(n)$ is called an *instrumental variable* (IV) if, (1) it is uncorrelated with the additive noise $w(n)$, that is, $E\{w(n)z(n+m)\} = 0$, $m > 0$; and (2) with $m = 1, \ldots, M \geq p$, we get a full rank set of linear equations from (1-01). The delayed process $z(n) = y(n-d)$ is an IV when $w(n)$ is MA(q_w) noise and $d > p + q_w$. Appropriate choices of $z(n)$ lead to estimates based on higher-order cumulants; for example, $z(n) = y(n)y(n-\rho)$, leads to estimates based on the 1-D slice $C_{3y}(m,\rho)$ of the third-order cumulant; other examples are discussed in [64]. For a discussion of *optimal* IV's, see [58]. When $z(n) \neq y(n)$, the resulting matrix is no longer Hermitian symmetric. A variation of RLS that is appropriate for this case is called the *Recursive Instrumental Variable* (RIV) algorithm, and is implemented in `rivtr`. The algorithm is given by:

Compute the weight vector $\mathbf{w}(i)$ recursively, as

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{z}(n)}{1+\lambda^{-1}\mathbf{y}^{H}(n)\mathbf{P}(n-1)\mathbf{z}(n)} \quad \textbf{(1-102)}$$

$$\alpha(n) = d(n) - \mathbf{w}^{H}(n-1)\mathbf{y}(n) \quad \textbf{(1-103)}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\alpha^{*}(n) \quad \textbf{(1-104)}$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{y}^{H}(n)\mathbf{P}(n-1) \quad \textbf{(1-105)}$$

where the desired signal d(n) = y(n + 1). The initial values are $\mathbf{w}(0) = 0$ and $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$, where δ is a small positive constant. Here $y(n)$ is the observed process, and $z(n)$ is the IV; when $z(n) = y(n)$, the RIV reduces to RLS.

Examples

```
load riv
ar(:,1)=rivtr(y,2);
ar(:,2)=rivtr(y,3);
ar(:,3)=rivtr(y,4);
ar(:,4)=rivtr(zw,4);
ar(:,5)=rivtr(zc,4);
```

Now, let us look at the five steady-state AR parameter estimates.

```
disp(ar)
```

1.000	1.000	1.000	1.000	1.000
-1.4789	-1.5075	-1.4931	-1.5354	-1.4686
0.7600	0.7910	0.7867	0.8248	0.7684

The data used in this example correspond to an AR(2) model, with AR parameter vector [1, –1.5, 0.8]. The vector y contains the noiseless signal. Additive white Gaussian noise was added to y to obtain the noisy signal zw with a SNR of 10 dB. Colored Gaussian noise, generated by passing a white Gaussian sequence through the AR filter [1,0,0.49], was added to y to obtain zc, also at a SNR of 10 dB.

RIV Algorithm: Double-Lattice Form

An order- and time-recursive solution of (1-94) and (1-95)leads to the double lattice algorithm, and is given below.

Starting with $n = 1$, evaluate (1-106) through (1-117) for $m = 1, 2, \ldots, M$, where M is the final desired order; then repeat with $n = 2$ and so on:

$$\Delta^{f}_{m-1}(n) = \lambda\Delta^{f}_{m-1}(n-1) + \frac{f^{*}_{m-1}(n)\tilde{b}_{m-1}(n-1)}{\gamma^{*}_{m-1}(n-1)} \quad \textbf{(1-106)}$$

$$\Delta^{b}_{m-1}(n) = \lambda\Delta^{b}_{m-1}(n-1) + \frac{b^{*}_{m-1}(n-1)\tilde{f}_{m-1}(n)}{\gamma^{*}_{m-1}(n-1)} \quad \textbf{(1-107)}$$

$$\Gamma_{f,m}(n) = -\frac{\Delta^{f}_{m-1}(n)}{B_{m-1}(n-1)} \quad \textbf{(1-108)}$$

$$\Gamma_{b,m}(n) = -\frac{\Delta^{b}_{m-1}(n)}{F_{m-1}(n)} \quad \textbf{(1-109)}$$

$$\mu(n) = -\frac{F^{*}_{m-1}(n)}{B^{*}_{m-1}(n-1)} \quad \textbf{(1-110)}$$

$$f_{m}(n) = f_{m-1}(n) + \Gamma^{*}_{f,m}(n)b_{m-1}(n-1) \quad \textbf{(1-111)}$$

$$b_{m}(n) = b_{m-1}(n-1) + \Gamma^{*}_{b,m}(n)f_{m-1}(n) \quad \textbf{(1-112)}$$

$$\tilde{f}_{m}(n) = \tilde{f}_{m-1}(n) + \mu(n)\Gamma_{b,m}(n)\tilde{b}_{m-1}(n-1) \quad \textbf{(1-113)}$$

$$\tilde{b}_{m}(n) = \tilde{b}_{m-1}(n-1) + \mu^{-1}(n)\Gamma_{f,m}(n)\tilde{f}_{m-1}(n) \quad \textbf{(1-114)}$$

$$F_m(n) = F_{m-1}(n) - \frac{\Delta^f_{m-1}(n)\Delta^b_{m-1}(n)}{B_{m-1}(n-1)} \quad \textbf{(1-115)}$$

$$B_m(n) = B_{m-1}(n-1) - \frac{\Delta^f_{m-1}(n)\Delta^b_{m-1}(n)}{F_{m-1}(n)} \quad \textbf{(1-116)}$$

$$\gamma_m(n-1) = \gamma_{m-1}(n-1) - \frac{\tilde{b}^*_{m-1}(n-1)b_{m-1}(n-1)}{B^*_{m-1}(n-1)} \quad \textbf{(1-117)}$$

Note that (1-106) through (1-117) are in the proper order required for successful evaluation.

The initial values for the algorithm are given by

$$\Delta^f_{m-1}(0) = \Delta^b_{m-1}(0) = 0 \quad \textbf{(1-118)}$$

$$F_{m-1}(0) = B_{m-1}(0) = \delta \ll 1 \quad \textbf{(1-119)}$$

$$f_0(n) = b_0(n) = y(n) \quad \textbf{(1-120)}$$

$$\tilde{f}_0(n) = \tilde{b}_0(n) = z(n) \quad \textbf{(1-121)}$$

$$F_0(n) = B_0(n) = \lambda F_0(n-1) + z(n)y^*(n) \quad \textbf{(1-122)}$$

$$\gamma_0(n) = 1 \quad \textbf{(1-123)}$$

The following discussion is adapted from [64], which also gives a derivation of the algorithm.

Equations (1-111)and(1-112) with the initial condition in (1-120)define a lattice structure excited by the observed process $y(n)$. Equations (1-113) and (1-114) with the initial condition in (1-121) define a lattice structure excited by the

associated process $z(n)$. The two lattices are coupled through the time-update equations for the Δ's, (1-106) and (1-107), and the order-update equation for the conversion factor, (1-117). When $y(n) = z(n)$, the two lattices collapse into one.

In the above, $f_m(n)$ and $b_m(n)$ are the mth order prediction errors for the observed process $y(n)$; $\tilde{f}_m(n)$ and $\tilde{b}_m(n)$ are the mth order prediction errors for the associated process $z(n)$; $\Gamma_{f,m}(n)$ and $\Gamma_{b,m}(n)$ are the reflection coefficients for the forward/backward prediction of $y(n)$, and may be interpreted as normalized cross-correlations between the forward prediction error in one lattice and the backward prediction error in the other; $\mu(n)$ is the scale factor for converting the reflection coefficients associated with $y(n)$ to those associated with $z(n)$; $F_m(n)$ is the correlation between the mth order forward prediction errors, $f_m(n)$ and $\tilde{f}_m(n)$; $B_m(n)$ is the correlation between the mth order backward prediction errors, $b_m(n)$ and $\tilde{b}_m(n)$; $\gamma_m(n)$ is the scale factor for converting the *a priori* prediction errors to their *a posteriori* values; its absolute value is bounded by unity; $\Delta^b_{m-1}(n)$ is the cross-correlation between the *a posteriori* backward prediction error, $b_{m-1}(n-1)$, and the *a priori* tilded forward prediction error, $\tilde{\eta}_{m-1}(n)$; and $\Delta^f_{m-1}(n)$ is the cross-correlation between $f_{m-1}(n)$ and $\tilde{\varphi}_{m-1}(n-1)$. Time-update equations for $F_m(n)$ and $B_m(n)$ are given by (1-115) and (1-116).

This algorithm is implemented in routine `rivdl`.

Examples

```
load riv
ar(:,1)=rivdl(y,2);
ar(:,2)=rivdl(y,3);
ar(:,3)=rivdl(y,4);
ar(:,4)=rivdl(zw,4);
ar(:,5)=rivdl(zc,4);
```

Now let's see the five AR parameter estimates:

```
disp(ar)
```

1.000	1.000	1.000	1.000	1.000
-1.4814	-1.5106	-1.4941	-1.5358	-1.4692
0.7623	0.7941	0.7876	0.8252	0.7690

The data used in this example correspond to an AR(2) model, with AR parameter vector [1, –1.5, 0.8]. The vector y contains the noiseless signal. Additive white Gaussian noise was added to *y* to obtain the noisy signal *zw* with a SNR of 10 dB. Colored Gaussian noise, generated by passing a white Gaussian sequence through the AR filter [1, 0, 0.49], was added to *y* to obtain *zc*, also at a SNR of 10 dB.

Summary

We have discussed several linear prediction problems. The Trench algorithm is implemented in `trench`: it computes the backward and forward AR prediction filters associated with a given cumulant slice. Routines `rivtr` and `rivdl` implement the transversal and double lattice version of the Recursive Instrumental Variable (RIV) method: these algorithms may be based upon the correlation or a slice of the third- or fourth-order cumulant, depending upon the *Instrumental Variable* (IV) used; `ivcal` can be used to compute the IV.

Harmonic Processes and DOA

In the harmonic retrieval problem, the observed data are of the form

$$y(n) = \sum_{k=1}^{p} \alpha_k \exp(j2\pi n f_k + j\phi_k) + w(n) = x(n) + w(n) \quad \textbf{(1-124)}$$

where $w(n)$ is additive noise; α_k's are the amplitudes, f_k's are the frequencies, and ϕ_k's are the phases. Additional assumptions are often made regarding the phases; the typical assumption is that the phases are independent random variables uniformly distributed over $[0, 2\pi]$. Such an assumption implies that multiple realizations are available, that is,

$$y_m(n) = \sum_{k=1}^{p} \alpha_k \exp(j2\pi n f_k + j\phi_{k,m}) + w_m(n) = x_m(n) + w_m(n) \quad \textbf{(1-125)}$$

where m denotes the realization number. If only a single realization is available, then the phases are nonrandom constants. Since $2\cos(\theta) = e^{j\theta} + e^{-j\theta}$, we note that a signal consisting of p real harmonics can be handled by the model in (1-125) with $2p$ frequencies, $\pm f_k$, $k = 1, \ldots, p$.

The problem is to determine p and then to estimate the frequencies, amplitudes and phases of each harmonic. Intuitively, we expect the power spectrum of $x(n)$ to consist of peaks at $f = f_k$, perhaps masked by the spectrum of the noise. The classical power spectrum estimators (Blackman-Tukey and Welch) can be used only if the frequencies are well-separated, that is, $\min |f_m - f_n| > 1/L$, where L is the length of the estimated autocorrelation sequence in the Blackman-Tukey method, or the block size in the Welch method; hence, parametric models are useful for short data lengths.

The model in (1-125) is similar to the model for the observed data in the array processing/direction of arrival (DOA) problem. This problem arises in various applications, such as radar, sonar, geophysics, and analysis of EEG and ECG signals. An array of N independent sensors provides samples of the data at different points in space. Data are collected simultaneously from the set of sensors; the set collected at a single instant of time is called a *snapshot*, and is a vector of N observations. The signal recorded at each sensor consists of sensor

noise as well as signals from p sources, assumed to be uncorrelated with one another. The signal recorded at the nth sensor and mth snapshot is

$$y_m(n) = \sum_{k=1}^{p} A_k s_k(m - \tau_{k,n}) + w_m(n),$$

where $s_k(m)$ is the signal emitted by the kth source, and $\tau_{k,n}$ is the signal delay (with respect to some arbitrary reference point) at the nth sensor. Typically, the sensors are assumed to be *isotropic*, and the source signals are assumed to be *plane waves* originating from the far-field of the array. The signals are assumed to be narrow-band, and as such they are characterized by a single frequency, f_o or a single wavelength $\lambda = c/f_o$, where c is the speed of signal propagation. The received signal can now be written as,

$$y_m(n) = \sum_{k=1}^{p} \alpha_k(m)\exp(j\phi(k,n)) + w_m(n) = \qquad m = 1, \ldots, M, \qquad n = 1, \ldots, N, \tag{1-126}$$

where $\alpha_k(m)$ is the amplitude of the kth source signal at snapshot m; and $\phi(k,n)$ is the phase delay of the kth source signal at the nth sensor, and depends upon the array geometry.

In a circular array, the sensors are assumed to be equally distributed in angle on the circumference of a circle with radius R. Let the bearings, θ_k, be measured with respect to the radius vector passing through a sensor. The phase delay is then given by [23]

$$\phi(k,n) = \frac{2\pi R}{\lambda}\left[\cos\left(\frac{2\pi n}{N} - \phi_k\right) - \cos\left(\frac{2\pi n}{N}\right)\right] \tag{1-127}$$

For a linear array, the bearing of the plane wave is defined as the angle between the plane wave front and the normal to the array. The phase delay is given by

$$\phi(k,n) = \frac{2\pi d_n}{\lambda}\sin(\phi_k) \tag{1-128}$$

where θ_k is the bearing of the kth source, and d_n is the location of the nth source with reference to some arbitrary origin. For a uniformly spaced linear array, with spacing d, we can set $d_n = nd$. The observed signals are thus given by,

$$y_m(n) = \sum_{k=1}^{p} \alpha_k(m) \exp\left(j 2\pi n \frac{d \sin(\phi_k)}{\lambda}\right) + w_m(n) \qquad m = 1, \ldots, M \tag{1-129}$$

Comparing (1-129) with (1-125), we see that the term $d\sin(\theta_k)/\lambda$ plays the role of frequency, and $\alpha_k(m)$ plays the role of the complex amplitude $\alpha_k \exp(j\phi_{k,m})$. Time samples, indexed by n in (1-125), correspond to spatial samples, or sensor numbers in (1-129). The realization number m in (1-125) is interpreted as the snapshot number in (1-129). In (1-124), we assume $0 < f_k < 1$ in order to ensure that there is no aliasing. Similarly, in (1-129), we assume that $\lambda \geq 2d$ to ensure that there is no aliasing; a common assumption is $\lambda = 2d$ (half-wavelength spacing).

Thus, a solution to the harmonic retrieval problem in (1-125) also applies to the DOA problem in (1-129). In the former, we estimate power spectra, as functions of frequency; in the latter, we estimate angular spectra, as functions of angle, θ.

The following table states the duality between the harmonic retrieval and the DOA problems:

Harmonic Retrieval	DOA
Realization	Snapshot
Time sample	Sensor sample
Frequency f	Angular frequency $d \sin(\theta)/\lambda$
Random phases	Random complex amplitudes
$0 < f < 1$	$d \leq \lambda/2$
Temporal correlation matrix	Spatial correlation matrix
Periodogram	Beamformer

The autocorrelation matrix $\Re_{xx}$ with (k,l) entry $R_{xx}(l-k) = E\{x^*(n)x(n+k-l)\}$, which appears in the harmonic retrieval problem, will be replaced by the

spatial correlation matrix, $\Re = E\{\mathbf{x}(\mathbf{n})\mathbf{x}(\mathbf{n})^H\}$, where $\mathbf{x}(n)$ is the nth snapshot, in the DOA problem. The FT of the spatial (temporal) correlation sequence is called the *beamformer* (periodogram).

The autocorrelation sequence of the signal in (1-125) is given by,

$$R_{yy}(\tau) = \sum_{k=1}^{p} |\alpha_k|^2 \exp j2\pi\tau f_k + R_w(\tau) \qquad \textbf{(1-130)}$$

which follows because the harmonics are uncorrelated (phases are mutually independent). Note that the autocorrelation sequence is not absolutely summable; however, the FT of $R_{yy}(\tau)$ exists, provided we are willing to use the Dirac delta functions defined by $\int g(f)\delta_D(f) := g(0)$ [5, p.17].

In principle, one can estimate the frequencies by picking peaks in the estimated power spectrum; the variance of this frequency estimator is of order $1/N^3$ [22], where N is the length of the time series. The power spectrum is limited by its poor frequency resolution, that is, frequency separation must be larger than $1/N$. In practice, this severely limits the applicability of the power spectrum as a direct estimator of the frequency.

Synthetics for the two problems can be generated via `harmgen` and `doagen`.

Resolution and Variance

If $x(n)$ consists of two complex harmonics, at frequencies f_1 and f_2, we expect to see two peaks in the power spectrum. The resolution of a power spectral density estimator is the smallest value of $|f_1 - f_2|$ that leads to two discernible peaks. The resolution of the classical power spectrum estimators (e.g., Welch and Blackman-Tukey) is of order $1/M$, where M is the effective window length. In contrast, the resolution of the periodogram is $1/N$, and since $M \leq$ N, the periodogram exhibits higher resolution than the Blackman-Tukey and Welch estimators. The choice of the window function dictates the resolution-variance tradeoff: if $W(f)$ has a broad main lobe, the power spectrum estimate will be smoother, the variance of the estimate will be smaller, and the resolution will be lower.

It should be noted that zero-padding the data (or the correlation sequence) does not improve the resolution, it only makes the spectral estimate denser (this may lead to improved accuracy in estimating the locations of well-separated peaks).

AR and ARMA Models

The basic idea, in the parametric estimators, is that the noiseless harmonic process in (1-124) obeys a self-driving AR(p) model, that is, [28],

$$\sum_{k=0}^{p} a(k)x(n-k) = 0, \tag{1-131}$$

where $a(0) = 1$, and the roots of the $A(z)$ polynomial are given by $z = e^{j2\pi f_k}$, $k = 1, \ldots, p$. Hence, we can use any standard algorithm to estimate the AR parameters; the frequencies can be determined from the roots of the estimated AR polynomial. The determination of p is now equivalent to the AR model order determination problem.

In the presence of additive noise, process $x(n)$ in (1-124) satisfies the special ARMA(p,p) model,

$$\sum_{m=0}^{p} a(m)y(n-m) = \sum_{m=0}^{p} a(m)w(n-m) \tag{1-132}$$

where $w(n)$ is additive noise, not the driving noise of an ARMA model. Now, we can use any of the *standard* ARMA parameter estimation techniques to estimate the $a(k)$'s.

Multiplying (1-132) by $y^*(n-k)$, and taking expectations, we obtain,

$$\sum_{m=0}^{p} a(m)R_{yy}(k-m) = \sigma_w^2 a(k) \tag{1-133}$$

where we have assumed that the additive noise is white; hence, with $k > p$, we obtain a set of equations from which we can estimate the $a(k)$'s; these equations are sometimes called the "extended normal" equations.

The analysis extends to the real harmonic case where we obtain an AR(2p) or ARMA(2p,2p) model, with roots at $z = e^{\pm j2\pi f_k}$, $k = 1, \ldots, p$. Since the roots are all on $|z| = 1$, it turns out that $a(k) = a(2p - k)$, $k = 0, \ldots, 2p$.

Pisarenko's Method

The $M \times M$ autocorrelation matrix of the process $y(n)$ in (1-124) may be written as,

$$R_{yy} = R_{xx} + \sigma_w^2 I = SDS^H + \sigma_w^2 I \qquad \textbf{(1-134)}$$

where S is a Vandermonde matrix with (m,n) element $\exp(-j2\pi(m-1)f_n)$, $m = 1, \ldots, p$; and D is the diagonal matrix of the sinusoidal powers, $D = \text{diag}(P_1, \ldots, P_p)$. Since the frequencies are assumed to be distinct, matrix S has full rank p ($M \geq$ p). It follows immediately that the p largest eigenvalues of the matrix R_{yy} are given by, $P_k + \sigma_w^2$, $k = 1, \ldots, p$; the remaining $M - p$ eigenvalues are all equal to σ_w^2. With $M = p + 1$, it follows from (1-133) that [48]

$$\Re_{yy}\mathbf{a} = \sigma_w^2 \mathbf{a}. \qquad \textbf{(1-135)}$$

The vector **a** is the eigenvector corresponding to the minimum eigenvalue of the $(p + 1) \times (p + 1)$ correlation matrix $\Re_{yy}$. Once **a** has been estimated, the frequencies can be estimated as the roots of the $A(z)$ polynomial. The minimum eigenvalue solution to (1-135) is known as Pisarenko's method, and is implemented in routines `harmest` and `doa`. In order to ensure positive-definiteness of matrix $\Re_{yy}$, it is essential to use biased sample estimates of the autocorrelation. For a signal consisting of p real harmonics, the order should be taken as $2p$.

The Pisarenko algorithm separates the eigenvectors and eigenvalues of the matrix $\Re_{yy}$ into two classes: those associated with the signal and those associated with the noise. Extensions of the idea have resulted in the high-resolution eigen-subspace methods, such as the *Min-Norm* and *MUSIC* algorithms.

Multiple Signal Classification (MUSIC)

Let $M > p$. The autocorrelation matrix $\Re_{yy}$ has a linearly independent (orthogonal) set of eigenvectors. Denote the eigenvalues and eigenvectors by λ_k and $\mathbf{v}_k$. Define the M element signal vector,

$$\mathbf{e}_k = e^{j2\pi f_k}[e^{-j2\pi f_k}, \ldots, e^{-j2\pi M f_k}]^T \quad \textbf{(1-136)}$$

The eigenvectors corresponding to the p largest eigenvalues are said to constitute the *signal subspace*; the remaining $M - p$ eigenvectors constitute the *noise subspace*. The signals, that is, the harmonics in (1-124), lie in the signal subspace and are orthogonal to the noise subspace, that is,

$$\mathbf{e_k}^H \mathbf{v}_i = 0, \qquad i = p+1, \ldots, M; \qquad k = 1, \ldots, p;$$

hence, with

$$\mathbf{e}(f) = e^{j2\pi f}[e^{-j2\pi f}, \ldots, e^{-j2\pi M f}]^T \quad \textbf{(1-137)}$$

one can search for the set of frequencies such that $\mathbf{e}^H(f)\mathbf{v}_k = 0$, $k = p + 1, \ldots, M$. When sample estimates of the autocorrelation are used, these relationships will hold only approximately. In MUSIC (Multiple Signal Classification) one determines the frequencies of the harmonics by looking for peaks in the *spectrum* defined by

$$P_{yy}(f) = \left[\sum_{k=p+1}^{M} w(k) \left| \mathbf{e}^H(f)\mathbf{v_k} \right|^2 \right]^{-1} \quad \textbf{(1-138)}$$

where $w(k) = 1$. Usually, the correlation matrix used in the FBLS method is used. When $w(k) = 1/\lambda_k$, we get the so-called Eigenvector solution, and when $w(k) = \delta(k - M)$, we get the Pisarenko solution. These algorithms are implemented in routines `harmest` and `doa`.

Minimum-Norm Method

The *minimum-norm* (Min-Norm) method is also an eigen-space method. Motivated by the normal equations for the harmonic signal model, one seeks the AR vector, **a**, which is orthogonal to the signal subspace (recall the discussion of the Pisarenko method). Let us decompose the signal subspace, as

$$Q_s = [\mathbf{v}_1, \ldots, \mathbf{v}_p] = \begin{bmatrix} \mathbf{x}_s{}^T \\ \mathbf{X}_s \end{bmatrix} \tag{1-139}$$

The condition $Q_s^T \mathbf{a} = \mathbf{0}$, can be rewritten as,

$$X_s^T \mathbf{w} = \mathbf{x}_s \tag{1-140}$$

where $\mathbf{w} = -[a(1), \ldots, a(p)]^T$. Equation (1-140) represents p equations in $M - 1$ unknowns. Hence, if $p < M - 1$, the set of equations is underdetermined, and the solution is not unique. A unique solution is obtained by choosing the Min-Norm solution, that is,

$$\mathbf{w} = [1 - \mathbf{x}_s^H \mathbf{x}_s]^{-1} X_s^* \mathbf{x}_s \tag{1-141}$$

This solution, called the *minimum-norm* solution, was developed by Kumaresan and Tufts and is implemented in routines `harmest` and `doa`.

Pisarenko's method uses a single noise subspace eigenvector and, hence, is not as robust as the general MUSIC estimator. The Min-Norm method has been reported to have a smaller bias than the MUSIC algorithm for estimating the frequencies. An overview of eigen-methods is given in [26].

A modification of the FBLS algorithm has been reported by Kumaresan and Tufts [23], where the correlation matrix $\tilde{\Phi}$ is replaced by throwing out the noise subspace eigenvectors, that is, if λ_k and $\mathbf{v}_k$ denote the eigenvalues and eigenvectors of $\tilde{\Phi}$, then,

$$\hat{\Phi} = \sum_{k=1}^{p} \lambda_k \mathbf{v}_k \mathbf{v}_k^H \tag{1-142}$$

This procedure of replacing Φ by Φ is sometimes called the low-rank SVD approximation. The algorithm then proceeds as in FBLS. Kumaresan and Tufts [29] have established that in the noiseless case, under the assumption, $p \le M \le N - p/2$, p of the roots of $A(z)$ lie on the unit circle, and correspond to the true frequencies; the remaining $(M - p)$ roots are uniformly distributed in angle inside the unit circle; these roots are called the *extraneous* zeros. High-resolution estimates are obtained by choosing large values for the predictor order, that is, $M \approx 3N/4$.

ESPRIT

Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) is an eigen-space method developed by Kung, Rao *et al* [30], and by Roy and Kailath [57]; however, the basic idea behind this algorithm is considerably different from that in MUSIC and Min-Norm methods.

In ESPRIT, the basic idea is to decompose the sensor array into two (possibly overlapping) subarrays, and then to use the cross-correlation between the two subarrays to estimate the source bearings. The discussion here is in terms of the DOA problem, but as we have seen earlier, the treatment is applicable to the harmonic retrieval problem as well.

Assume that we have two arrays, such that elements of one array are at a constant displacement Δ with respect to corresponding elements of the other array. For example, if we have an uniformly spaced linear array of N elements, we can partition this array into two subarrays, each consisting of L elements, with a displacement of $\Delta = N - L$ between the two subarrays, that is, with sensors $1, \ldots, N - \Delta$ in the first subarray, and sensors $\Delta + 1, \ldots, N$ in the second array.

Let $y_m(n)$ and $z_m(n)$, $n = 1, \ldots, L$, denote the signals at the elements of the two subarrays, that is,

$$y_m(n) = \sum_{k=1}^{p} \alpha_k(m) \exp(j\phi(k, n)) + w_m(n) \qquad \textbf{(1-143)}$$

$$z_m(n) = \sum_{k=1}^{p} \alpha_k(m) \exp(j\phi(k, n)) e^{j2\pi\Delta d \sin(\phi_k)/\lambda} + v_m(n) \qquad \textbf{(1-144)}$$

The mth snapshot can be written in vector form as,

$$\mathbf{y}_m = A\mathbf{a}_m + \mathbf{w}_m \tag{1-145}$$

$$\mathbf{z}_m = A\Phi\mathbf{a}_m + v_m \tag{1-146}$$

where Φ is a diagonal matrix with (k,k) entry, $\exp(j2\pi\Delta d\sin(\theta_k)/\lambda)$; matrix A has (n,k) entry, $\exp(j\phi(k,n))$; $\mathbf{a}_m = [\alpha_1(m), \ldots, \alpha_M(m)]^T$ is the source signal vector at the mth snapshot; and, $\mathbf{w}_m$ and $\mathbf{v}_m$ are the noise vectors. The bearing information can be recovered from matrix Φ.

The auto- and cross-correlation matrices of $\mathbf{y}(n)$ and $\mathbf{z}(n)$ are given by,

$$\Re_{yy} = E\left\{\mathbf{y}_m\mathbf{y}_m^H\right\} = ASA^H + \sigma^2 I_L \tag{1-147}$$

$$\Re_{yz} = E\left\{\mathbf{y}_m z_m^H\right\} = AS\Phi^H A^H + \sigma^2 J_L \tag{1-148}$$

where we have assumed that the additive noise is white (spatially uncorrelated from sensor to sensor), and has variance σ^2.

Matrix $S = E\left\{\mathbf{a}_m\mathbf{a}_m^H\right\}$ is the signal correlation matrix and is nonsingular if the sources are incoherent.

The structure of matrix J_L depends upon the common elements of the two subarrays. If the two subarrays have no common elements, then J_L is the zero matrix. In the uniform linear array example considered earlier, $z_m(n) = y_{m+\Delta}(n)$, $m = 1, \ldots, L - \Delta$, and

$$J_L = \begin{bmatrix} \mathbf{0}_{\Delta\times L} & \mathbf{0}_{\Delta\times\Delta} \\ \mathbf{I}_{L-\Delta} & 0_{L-\Delta\times\Delta} \end{bmatrix}$$

In any case, given the knowledge of the geometries of the two arrays, we can compute matrix J_L.

We can estimate σ^2 as the smallest eigenvalue of matrix $\Re_{yy}$. We can now eliminate the noise contribution, and create the signal correlation matrices,

$$C_{yy} = \Re_{yy} - \sigma^2 I_L = ASA^H \quad \textbf{(1-149)}$$

$$C_{yz} = \Re_{yz} - \sigma^2 I_L = AS\Phi^H A^H \quad \textbf{(1-150)}$$

Given two $n \times n$ matrices, A and B, the generalized eigenvalues of (A,B), are defined by [21]

$$\lambda(A, B) := \{z : det(A - zB) = 0\}.$$

If B is nonsingular, there are exactly n generalized eigenvalues, and these are identical with the eigenvalues of $B^{-1}A$. The singular B case is discussed in [21].

It is shown in [30, 57] that the generalized eigenvalues of (C_{yy},C_{yz}) are given by $\lambda_k = \exp(j2\pi\Delta d\sin(\theta_k)/\lambda)$, $k = 1, \ldots, p$. Once λ_k has been estimated, we can readily obtain the θ_k's. This algorithm — for a uniformly spaced linear array — is implemented in routine `doa`; the generalized eigenvalues are computed via the MATLAB built-in function `eig`. Algorithms based on fourth-order cumulants are discussed next.

Criterion-Based Estimators

The FT, at frequency ω_o, can be viewed as a filter that passes only the components with frequency ω_o, while suppressing the rest. With finite data lengths, however, the spectral estimate suffers from the problems of sidelobe leakage (with details dictated by the choice of window function).

Consider the length p finite impulse response (FIR) filter,

$$y(n) = \sum_{k=0}^{p} a(k)x(n-k) \quad \textbf{(1-151)}$$

where the input $x(n)$ is assumed to be zero mean. The variance of the filter output is given by,

$$\sigma_y^2 = E\{|y(n)|^2\} = \mathbf{a}^H R_{xx}\mathbf{a} \tag{1-152}$$

where $\mathbf{a} = [a(0), \ldots, a(p)]^T$, and R_{xx} is the $(p + 1) \times (p + 1)$ autocorrelation matrix.

We would like the output variance σ_y^2 to be a measure of the power spectral density of $x(n)$ at frequency f_o. Alternatively, the FIR filter should pass a sinusoid at frequency f_o with unity gain; hence, we have the constraint,

$$\sum_{k=0}^{p} a(k)\exp(-j2\pi k f_o) = 1. \tag{1-153}$$

We also want to minimize the contribution due to sinusoids at frequencies other than f_o; this can be achieved by minimizing the output variance in (1-152) subject to the constraint in (1-153). The resulting estimator is given by

$$P_{ml}(f) = [\mathbf{e}^H(f) R_{xx}^{-1} \mathbf{e}(f)]^2 \tag{1-154}$$

where $\mathbf{e} = [1, e^{-j2\pi f}, \ldots, e^{-j2\pi p f}]^H$. This estimator, which is variously called Capon's maximum-likelihood estimator or the Minimum Variance Distortionless estimator [8, 27], is not a true power spectral density estimator. For example, if $x(n)$ consists of a harmonic at frequency f_o and power P that is observed in white noise with variance σ_w^2, then,

$$P_{ml}(f_o) = P + \sigma_w^2/p,$$

where p is the number of autocorrelation lags used; note that the noise power contribution has been reduced by a factor of p. The resolving power of this estimator is reported to be [27]

$$\Delta f \geq \frac{1}{p\sqrt{SNR}}$$

where SNR is the signal-to-noise ratio (ratio of signal variance to noise variance), and p is the length of the autocorrelation sequence. This estimator is implemented in routines `harmest` and `doa`.

The maximum-entropy spectral estimator of Burg is based on the availability of *exactly known* autocorrelation lags, $\{R_{xx}(m)\}_{m=0}^{p}$, (in which case a Gaussian pdf and an AR(p) model are obtained for process $x(n)$). Routine `trench` may be used to estimate the AR model parameters. The resolving power is reported to be

$$\Delta f \geq \frac{1}{pSNR}.$$

For SNR ≥ 1, the maximum entropy estimator has better resolving power than does Capon's maximum-likelihood estimator, which in turn is better than the Blackman-Tukey estimator.

Cumulant-Based Estimators

In [66] it was established that the fourth-order cumulants of the process in (1-124) are given by

$$C_{4y}(l, m, n) = -\sum_{k=1}^{p} e^{j2\pi f_k(l+m-n)}\left|\alpha_k\right|^4 + C_{4w}(l, m, n) \qquad \textbf{(1-155)}$$

In particular, the diagonal slice is given by,

$$C_{4y}(m, m, m) = -\sum_{k=1}^{p} e^{j2\pi f_k m}\left|\alpha_k\right|^4 + C_{4w}(m, m, m) \qquad \textbf{(1-156)}$$

If $w(n)$ is Gaussian, then $C_{4w}(m,m,m) \equiv 0$. If $w(n)$ is i.i.d. non-Gaussian, then $C_{4w}(m,l,n)$ is a delta function at the origin. Note that when $w(t)$ is Gaussian or i.i.d. non-Gaussian, (1-156) is similar to the autocorrelation sequence of the noiseless signal. Consequently, all of the analyses based on the autocorrelation carry over to the fourth-order cumulant. In particular, the development of the Eigenvector, Pisarenko, ML-Capon, AR, MUSIC, Min-Norm, and beamformer *spectral* estimates based on second-order statistics, can also be based on fourth-order statistics [44]. Similarly we can base ESPRIT on the fourth-order cross-cumulant matrix as well. The fourth-order statistics are most useful when the additive noise is narrow-band Gaussian.

Routine `harmest` implements algorithms for the harmonic retrieval problem: it estimates the spectrum using the Eigenvector, MUSIC, Pisarenko, ML, and AR algorithms based on the diagonal slice of the fourth-order cumulant.

Routine doa implements algorithms for the direction of arrival problem: it estimates the *angular spectrum* using the Eigenvector, MUSIC, Pisarenko, ML-Capon, ML, and AR algorithms based on the diagonal slice of the fourth-order cumulant. Routine doa implements the corresponding algorithms based on second-order statistics; it also implements ESPRIT.

Examples

```
load harm
Pxx=harmest(zmat,12,4,'unbiased',256,4);
```

You should see the display in Figure 1-12.

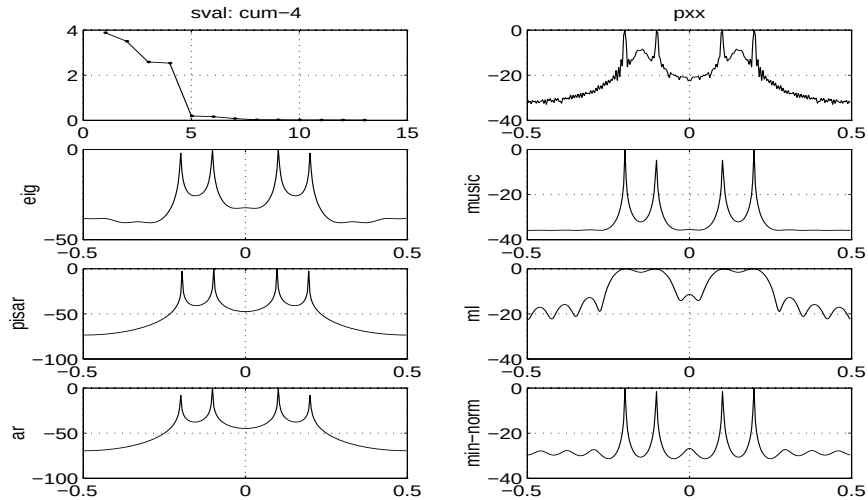


Figure 1-12 Singular Values of Cumulant Matrix and Estimated Spectra (harmest)

The data consist of two unit amplitude real harmonics at frequencies 0.1 Hz and 0.2 Hz, and are contaminated by colored Gaussian noise, with a variance of 0.5; the noise spectrum has a pole at 0.15 Hz, with a damping factor of 0.9. The SVD plot is indicative of two real harmonics, and the spectrum appears to peak at the correct frequencies, 0.1 and 0.2 Hz. The fourth-order cumulants do not see the Gaussian process at 0.15 Hz. Note that the power spectra are displayed in dB scale, that is, $10\log_{10}(\mathbf{Pxx})$

```
Pxx = harmest(zmat,12,6,'unbiased',256,2);
```

You should see the display in Figure 1-13. The SVD plot is indicative of three real harmonics, and the spectrum appears to peak at the correct frequencies, 0.1 and 0.2 Hz, as well as at 0.15 Hz, which is due to the Gaussian noise.

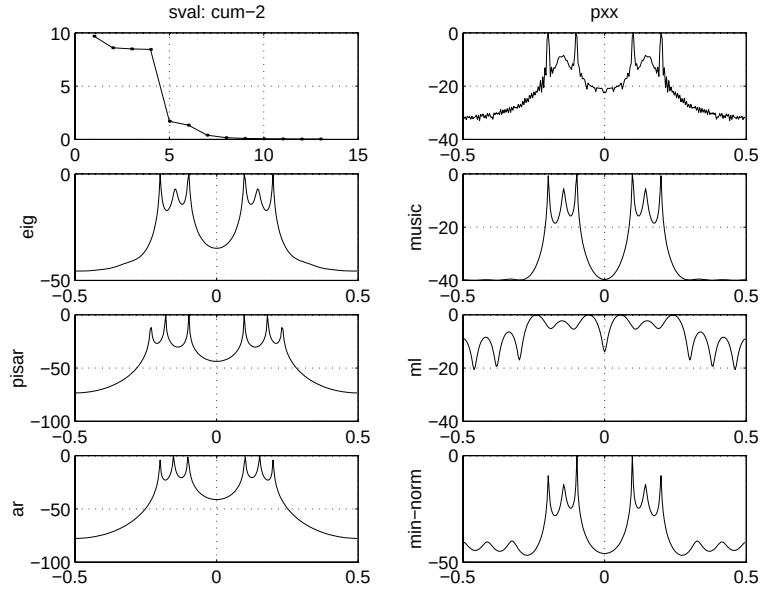


Figure 1-13 Singular Values of Correlation Matrix, and Estimated Spectra (harmest)

Examples

```
load doa1
[spec,theta]=doa(ymat,0.5,1,3,2,1);
```

You should see the display in Figure 1-14; note that the angular spectra are plotted in dB scale, that is, $10\log_{10}(\mathbf{spec})$.

The singular value plot indicates the possible presence of three sources. The observed signal consists of signals from two sources at –15 and –25 degrees, and is contaminated with spatially correlated noise that acts as a virtual source at a bearing of 30 degrees. The bearings estimated by ESPRIT should be –15.4634, –25.8851 and 29.9855. As expected, the Eigenvector, ML, AR, MUSIC and Min-Norm methods resolve the two sources and the virtual source due to the noise, whereas the beamformer does not. The MUSIC and Min-Norm estimates are virtually identical in this example. The vertical lines and circles denote the true bearings; the cross denotes the virtual bearing of the spatially correlated noise.

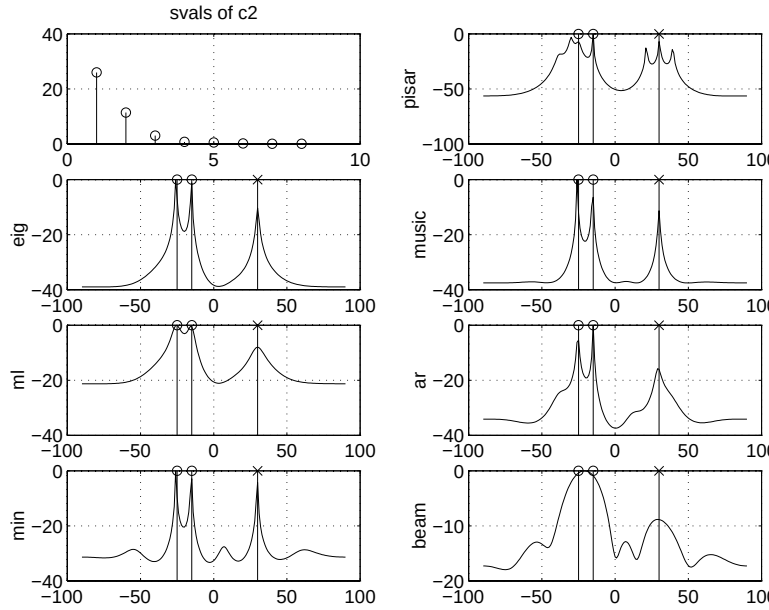


Figure 1-14 Singular Values of Spatial Correlation Matrix, and Estimated Angular Spectra (doa)

```
load doa1
[spec,theta]=doa(ymat,0.5,1.2);
```

The singular value plot indicates the possible presence of two sources; the estimated angular spectra show peaks at around the true bearings of –15 and –25 degrees; notice that the spatially correlated Gaussian noise source has been virtually suppressed by using fourth-order cumulants. The vertical lines and circles denote the true bearings; the cross denotes the virtual bearing of the spatially correlated noise. The bearings estimated by ESPRIT should be –15.0066 and –25.1361.

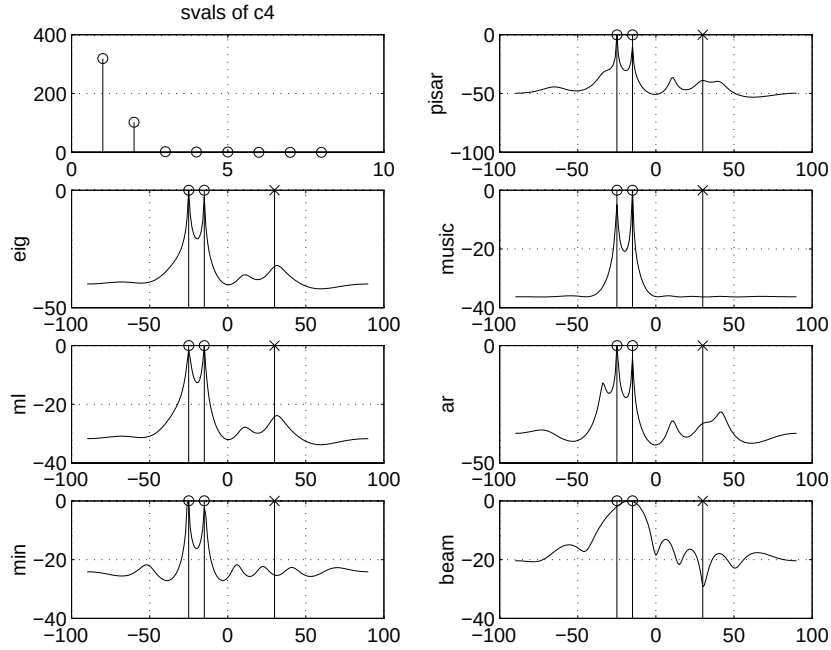


Figure 1-15 Singular Values of Spatial Cumulant Matrix, and Estimated Angular Spectra (doa)

Summary

The estimation of the frequencies of harmonics observed in noise, or that of determining the bearings of sources in the DOA problem can be treated from either a parametric or a nonparametric view point. In the latter case, we obtain nonparametric estimates of spectra and angular spectra. In the former case, we have several high-resolution algorithms.

Routine `harmest` can be used for the harmonic retrieval problem; the user has the choice of using second- or fourth-order cumulants. This routine estimates power spectra using the MUSIC, Eigenvector, Pisarenko, ML (Capon), AR and Min-Norm methods, based on either the diagonal slice of fourth-order cumulants, or on the covariance; it also estimates the conventional periodogram. The number of harmonics can usually be determined by examining the singular value plot generated by `harmest`.

Routine `doa` can be used to solve the direction-of-arrival problem (DOA) for a uniformly spaced linear array. It estimates angular spectra using the Beamformer, ML-Capon, AR, MUSIC, Pisarenko's method, eigenvector, Min-Norm and ESPRIT algorithms, based on the spatial cross-correlation or cross-cumulant. Routine `pickpeak` may be used to pick peaks in the estimated (angular) spectra.

Nonlinear Processes

The simplest nonlinear system is the second-order Volterra system whose input-output relationship is defined by

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) + \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} q(k,l)x(n-k)x(n-l) \quad \textbf{(1-157)}$$

The corresponding frequency domain representation is

$$Y(f) = H(f)X(f) + \sum_{f_1+f_2=f} Q(f_1,f_2)X(f_1)X(f_2) \quad \textbf{(1-158)}$$

and is obtained by Fourier Transforming both sides of (1-157). The time-domain product term $x(n-k)x(n-l)$ leads to convolution in the frequency domain, which is represented by the condition $f = f_1 + f_2$. It is usually assumed that the quadratic kernel is real and symmetric, that is, $q(k,l) = q(l,k)$, or equivalently, $Q(f_1,f_2) = Q(f_2,f_1) = Q^*(-f_1,-f_2)$. It is readily verified that $Q(f_1,f_2)$ in the region $|f_2| \le f_1, 0 \le f_1 \le 1/4$, specifies $Q(f_1,f_2)$ everywhere.

The basic problem is given $x(n)$ and $y(n)$, $n = 1, \ldots, N$, we want to estimate the linear part, $h(k)$, and the quadratic part, $q(k,l)$.

Solution Using Cross-Bispectra

In [70], an algorithm to estimate the parameters of the model in (1-157) was developed, under the assumption that $x(n)$ is stationary Gaussian. From (1-157), we can see that the cross-spectrum is given by,

$$S_{yx}(f) = E\{Y(f)X^*(f)\} = H(f)E\{X(f)X^*(f)\} = H(f)S_{xx}(f) \quad \textbf{(1-159)}$$

The term involving the triple product of the $X(f)$'s disappears since $x(n)$ is symmetrically distributed. Since $S_{xx}(f)$ and $S_{xy}(f)$ can be estimated, we can estimate $H(f)$, the linear part.

Consider the cross-cumulant

$$C_{xxy}(\tau, \rho) = E\{x(n)x(n+\tau)y(n+\rho)\} \tag{1-160}$$

$$= E\{x(n)x(n+\tau)\sum_{k=0}^{\infty} h(k)x(n+\rho-k) + \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} q(k,l)x(n+\rho-k)x(n+\rho-l)\}$$

$$= \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} q(k,l)E\{x(n)x(n+\tau)x(n+\rho-k)x(n+\rho-l)\} \tag{1-161}$$

which follows since $x(t)$ is symmetrically distributed. Assuming Gaussianity, we can rewrite the last equation in the frequency domain, as

$$S_{yxx}(f_1, f_2) = 2Q(f_1, f_2)S_{xx}(f_1)S_{xx}(f_2) + S_{xx}(f_2)\delta(f_1 + f_2)E\{y(n)\} \tag{1-162}$$

which is obtained under the assumption that $q(k,l) = q(l,k)$. The cross-bispectrum, $S_{yxx}(f_1,f_2)$ can be estimated via routine `bispecdx` and the spectra can be estimated via routine `spectrum`; we can then estimate the quadratic transfer function, $Q(f_1,f_2)$ from (1-162). This algorithm is implemented in routine `nltick` [70]. Note that the input process is assumed to be Gaussian.

Examples

```
load nl1
[h,q] = nltick(x,y,128,1);
```

You should see the display on Figure 1-16. A description of the signals `x` and `y` in `nl1.mat` is given in the section "Data Files." The true linear and quadratic impulse responses are shown in Figure 1-51.

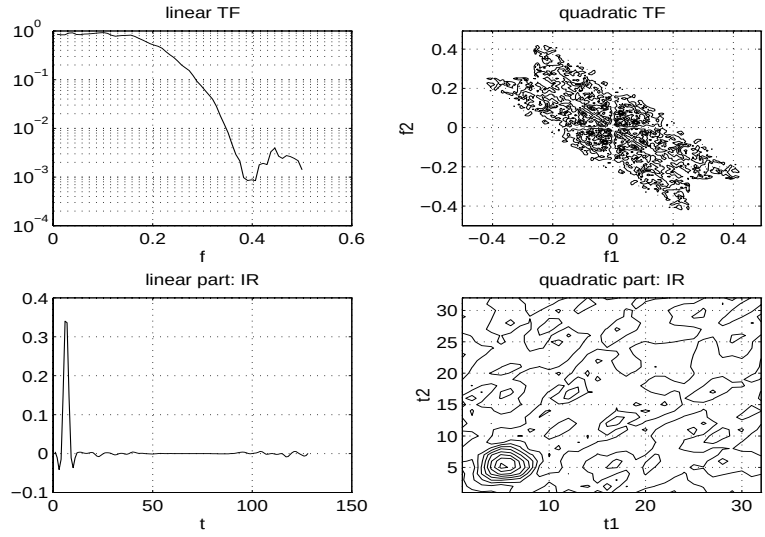


Figure 1-16 Transfer Functions Estimated by nltick

Solution Using FTs

In [50], another algorithm to estimate the parameters of the model in (1-157) was developed, without assuming that $x(n)$ is Gaussian; indeed, $x(n)$ may be deterministic. Given the input and output sequences, $x(n)$ and $y(n)$, we can compute their FTs, $X(f)$ and $Y(f)$ on a suitable grid, $f_m = m/M$, where M is the FFT length. We can collect (1-158) at the various frequencies into a set of equations that are linear in the unknowns, namely $H(f)$ and $Q(f_1,f_2)$ [50].

Let $r = m \bmod 2$, and let $m_+ := (m + r)/2$, $m_- := (m - r)/2$, . For $m = 0, \ldots, M/2$, let the unknown parameter vector be

$$\mathbf{b}(m) = \left[H(m), Q(m_+, m_-), Q(m_+ + 1, m_- - 1), \ldots, Q\left(\frac{M}{4}, m - \frac{M}{4}\right)\right]^T$$

and,

$$A = \left[X(m), X(m_+)X(m_-), X(m_+ + 1)X(m_- - 1), \ldots, X\left(\frac{M}{4}\right)X\left(m - \frac{M}{4}\right)\right]^T$$

Then, we have the system of linear equations,

$Y(m) = \mathrm{A}^T\mathbf{b}(m)$,

which is overdetermined; we can obtain the least-squares solution for each frequency grid point, m.

This algorithm is implemented in routine `nlpow`.

Examples

```
load nl1
[h1,q1]=nlpow (x,y,128);
```

You should see the display on Figure 1-17.

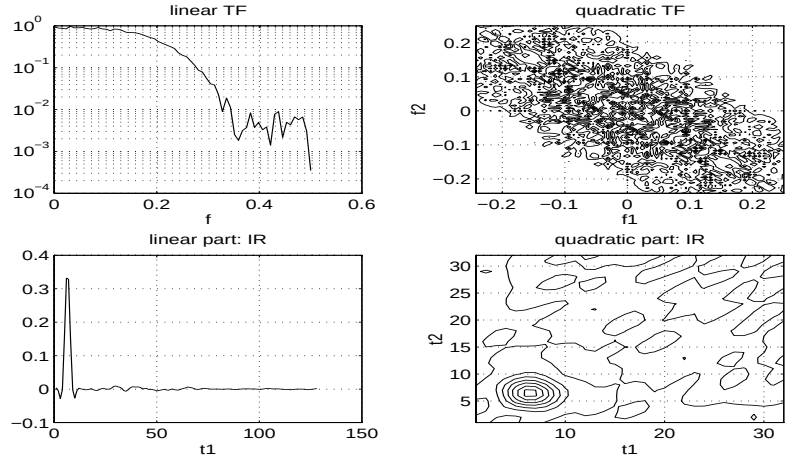


Figure 1-17 Transfer Functions Estimated by nlpow

```
load nl2
[h2,q2]=nlpow (x,y,128);
```

You should see the display on Figure 1-18.

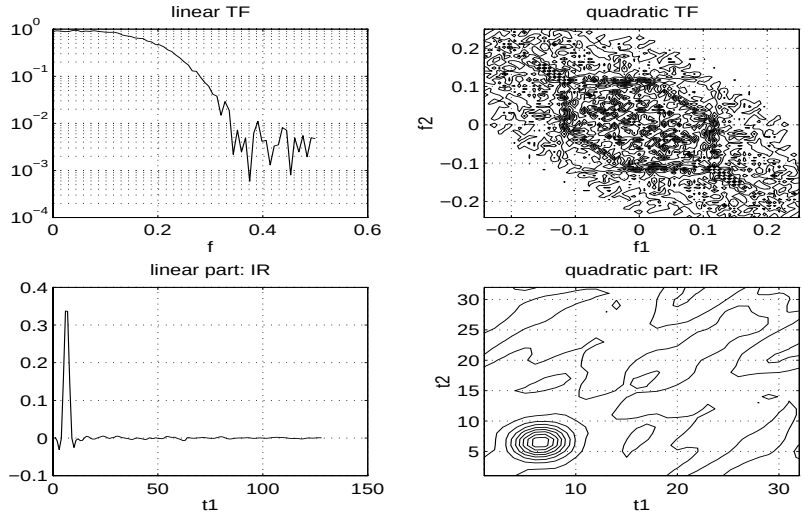


Figure 1-18 Transfer Functions Estimated by nlpow

As expected, the plots in the two figures are quite similar. A description of the signals **x** and **y** in the files `nl1.mat` and `nl1.mat` may be found in the section ““Data Files.” The true linear and quadratic impulse responses are shown in Figure 1-51.

Quadratic Phase Coupling

Phase coupling occurs due to nonlinear interactions between harmonic components. Three harmonics with frequencies f_k and phases ϕ_k, k = 1,2,3, are said to be quadratically phase coupled if $f_3 = f_1 + f_2$ and $\phi_3 = \phi_1 + \phi_2$. Quadratic phase coupling (coupling at sum and difference frequencies) occurs when a signal is passed through a square-law device, for example, and may be detected from the bispectrum [53].

Consider the signal $x(n)$, which is a mixture of harmonics with independent phases, and quadratically phase coupled sinusoids, that is,

$$x(n) = \sum_{i=1}^{N} \alpha_i \cos(2\pi f_i n + \phi_i) + \sum_{i=1}^{N_q} \sum_{k=1}^{N} \alpha_{ik}^q \cos(2\pi f_{ik}^q n + \phi_{ik}^q) \quad \textbf{(1-163)}$$

where $f_{i3}^q = f_{i1}^q + f_{i2}^q$, $\phi_{i3}^q = \phi_{i1}^q + \phi_{i2}^q$, the f's are all distinct, and ϕ_{i1}^q, ϕ_{i2}^q, and ϕ_i are all i.i.d. and uniformly distributed over $[-\pi,\pi]$.

Then, the third-order cumulant of $x(n)$ is given by [66]

$$\begin{aligned} C_{3x}(\tau_1, \tau_1) &= \frac{1}{4} \sum_{i=1}^{N_q} (\alpha_{i1}^q \alpha_{i2}^q \alpha_{i3}^q) \times \\ &[\cos(2\pi f_{i1}^q \tau_1 + 2\pi f_{i2}^q \tau_2) + \cos(2\pi f_{i1}^q \tau_1 - 2\pi f_{i3}^q \tau_2) \\ &+ \cos(2\pi f_{i2}^q \tau_1 - 2\pi f_{i3}^q \tau_2) + \cos(2\pi f_{i2}^q \tau_1 + 2\pi f_{i1}^q \tau_2) \\ &+ \cos(2\pi f_{i3}^q \tau_1 - 2\pi f_{i1}^q \tau_2) + \cos(2\pi f_{i3}^q \tau_1 + 2\pi f_{i2}^q \tau_2)] \end{aligned} \quad \textbf{(1-164)}$$

The diagonal slice, $C_{3x}(\tau,\tau)$, is given by

$$C_{3x}(\tau, \tau) = \frac{1}{2} \sum_{i=1}^{N_q} \prod_{j=1}^{3} \alpha_{ij}^q \sum_{k=1}^{3} \cos(2\pi f_{ik}^q \tau) \quad \textbf{(1-165)}$$

The 2-D FT of (1-164) yields the bispectrum,

$$
\begin{aligned}
&S_3(\lambda_1, \lambda_2) \\
&\quad = \frac{1}{8} \sum_{i=1}^{N_q} (\alpha_{i1}^q \alpha_{i2}^q \alpha_{i3}^q) \\
&\quad [\delta(\lambda_1 + f_{i1}^q)\delta(\lambda_2 + f_{i2}^q) + \delta(\lambda_1 + f_{i2}^q)\delta(\lambda_2 + f_{i1}^q) \\
&\quad +\delta(\lambda_1 - f_{i1}^q)\delta(\lambda_2 - f_{i2}^q) + \delta(\lambda_1 - f_{i2}^q)\delta(\lambda_2 - f_{i1}^q) \\
&\quad +\delta(\lambda_1 + f_{i1}^q)\delta(\lambda_2 - f_{i3}^q) + \delta(\lambda_1 - f_{i3}^q)\delta(\lambda_2 + f_{i1}^q) \\
&\quad +\delta(\lambda_1 - f_{i1}^q)\delta(\lambda_2 + f_{i3}^q) + \delta(\lambda_1 + f_{i3}^q)\delta(\lambda_2 - f_{i1}^q) \\
&\quad +\delta(\lambda_1 + f_{i2}^q)\delta(\lambda_2 - f_{i3}^q) + \delta(\lambda_1 - f_{i3}^q)\delta(\lambda_2 + f_{i2}^q) \\
&\quad +\delta(\lambda_1 - f_{i2}^q)\delta(\lambda_2 + f_{i3}^q) + \delta(\lambda_1 + f_{i3}^q)\delta(\lambda_2 - f_{i2}^q)]
\end{aligned}
\tag{1-166}
$$

It is evident that the nonredundant region of the bispectrum shows peaks only at the phase and frequency coupled bifrequencies, (f_{i1}^q, f_{i2}^q). The FT of the diagonal slice in (1-165), on the other hand, displays peaks at each of the three frequencies involved in the phase-coupling.

Consider () with $N_q = 1$. We have, omitting unnecessary superscripts,

$$
C_{3x}(\tau, \tau) = \frac{1}{2}\alpha_1\alpha_2\alpha_3 \sum_{k=1}^{3} \cos(2\pi f_k \tau)
\tag{1-167}
$$

where $f_3 = f_2 + f_1$. Note that the diagonal slice of the third-order cumulant is expressed as the sum of three harmonics. From our discussions on AR models for harmonics, we note that $C_{3x}(\tau,\tau)$ satisfies a self-driven AR(6) model, whose roots are at $\exp(\pm j2\pi f_k)$, $k = 1, 2, 3$. If we estimate the AR polynomial, we can compute the parametric bispectrum

$$
S_{3x}(\eta_1, \eta_2) = A(\eta_1)A(\eta_2)A^*(\eta_1 + \eta_2)
\tag{1-168}
$$

The parametric bispectrum estimator is implemented in routine `qpctor`.

Examples

```
load qpc
[arvec,bspec] = qpctor(zmat,18,12);
Maximum of bispectrum: B(0.1484,0.1016) = 4239
```

You should see the display on Figure 1-19. Note that $f_1 + f_2 \approx 0.25$, which corresponds to the third peak in the amplitude spectrum, so that we may conclude that three of the four harmonics are quadratically phase coupled. The singular value plot shows six significant singular values corresponding to one quadratically coupled triplet; as in the case of the power spectrum (harmonics in noise), overestimating the number of harmonics usually leads to better results; in this case, we used an AR order of 12.

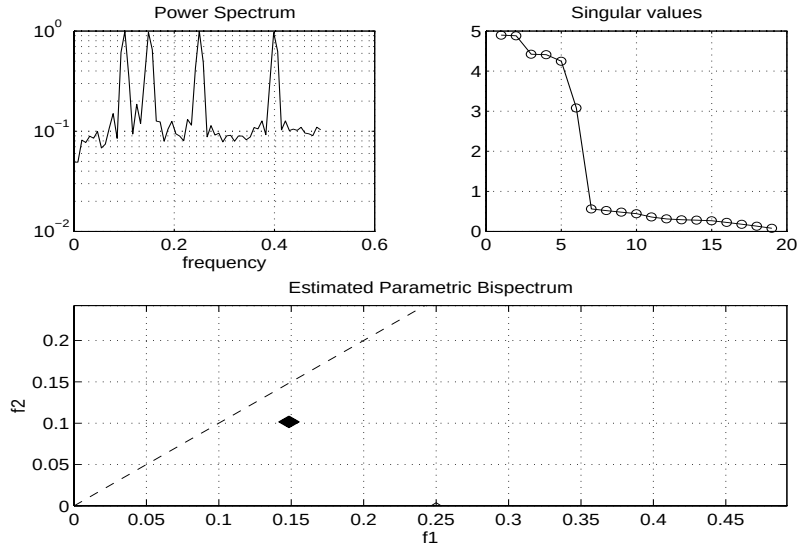


Figure 1-19 Parametric Estimate of bispectrum (qpctor)

Ideally, $A(\eta)$ contains impulses at $\pm f_k$, k = 1,2,3, and $S_{3x}(\eta_1,\eta_2)$ is given by

$$S_{3x}(\eta_1, \eta_2) = \left(\sum_{k=1}^{3} \delta\langle \eta_1 \pm f_k \rangle \right) \left(\sum_{k=1}^{3} \delta\langle \eta_2 \pm f_l \rangle \right) \left(\sum_{m=1}^{3} \delta\langle \eta_1 + \eta_2 \pm f_m \rangle \right) \quad \textbf{(1-169)}$$

From (1-166), we note that it suffices to examine the bispectrum in the region $0 \le \eta_2 \le \eta_1 \le 1/2$. If $f_2 \neq 2f1$, the parametric bispectrum will show a single impulse at (f_1,f_2), corresponding to $k = 1$, $l = 2$, $m = 3$ in (1-169), indicating the existence of the quadratically phase-coupled triplet $(f_1,f_2,f_1 + f_2)$. If $f_2 = 2f_1$, the bispectrum will have an additional peak at $(\eta_1,\eta_2) = (f_1,f_1)$, corresponding to $k = 1$, $l = 1$, $m = 2$ in (1-169), which can also result from the phase-coupled triplet $(f_1,f_1,2f_1)$.

It is important to note that the process in (1-163) is not ergodic, that is, consistent estimates of the third-order cumulant, $C_{3x}(\tau_1,\tau_2)$, cannot be obtained from a single realization, unless frequency-coupling is always accompanied by phase coupling [68]. Indeed, the random phase assumption itself suggests that multiple realizations are required. Further, creating multiple realizations by record segmentation does not lead to consistent estimates. Given a single realization, $C_{3x}(\tau_1,\tau_2)$ will show impulses if frequency coupling exists, that is, $f_3 = f_2 + f_1$. Given multiple realizations, $C_{3x}(\tau_1,\tau 2)$ (and $C_{3x}(\tau,\tau)$) will be nonzero only if both frequency and phase coupling exist, that is, $f_3 = f_2 + f_1$, and $\phi_3 = \phi_2 + \phi_1$.

The parametric bispectrum in (1-168) shows a peak whenever $A(\eta_1)$, $A(\eta_2)$, and $A(\eta_1 + \eta 2)$ have peaks. Hence, when $N_q > 1$, spurious peaks might result from interactions between different triplets of the form, $f_{il}^q + f_{jm}^q = f_{kn}^q$.

In the case of $N_q > 1$, interactions may occur between different QPC triplets leading to additional peaks in the parametric bispectrum given by (1-168).

Summary

The Higher-Order Spectral Analysis Toolbox offers two algorithms to estimate the parameters of a second-order Volterra model: `nltick`, which assumes that the input is Gaussian, and `nlpow`, which is applicable to arbitrary inputs. Note that both algorithms require access to inputs and outputs. Quadratic-phase coupling (QPC) can be detected using routine `qpctor`. Routine `qpcgen` can be used to generate synthetics for the QPC problem, and `nlgen` can be used to compute the output of a second-order Volterra system.

Time-Frequency Distributions

The popular and natural tool in linear system analysis is the Fourier transform, which decomposes a signal into its frequency components. The power spectrum (or energy spectrum) gives us information about the frequency components in the data, but not about the temporal localization of these components. Alternatively, the original time signal itself gives us good time resolution (obviously!) but does not give us any idea about frequency localization, except in trivial cases. So far, we have defined spectra, bispectra, and trispectra, only for stationary processes; as such, they are inappropriate for the analysis of nonstationary processes or transient signals. For nonstationary processes, we can define time-varying cumulants and polyspectra (in the same way that one talks about time-varying covariances and power spectra). Consider the signal, $y(t) = s(t) + g(t)$, where $s(t)$ is a deterministic signal and $g(t)$ is stationary zero-mean noise (perhaps Gaussian). The process $y(t)$ is nonstationary: its mean is $s(t)$, and for $k > 1$, $c_{ky} = c_{kg}$; thus, only the first-order cumulant carries information about the signal $s(t)$. In contrast, the higher-order moments of $y(t)$ depend both upon $s(t)$ and $g(t)$; thus, in this case, it is preferable to use higher-order moments [63]. For such signals, time-frequency distributions that describe the temporal evolution of the spectrum or the polyspectrum (moment spectrum) are useful tools.

A time-frequency distribution (TFD) is a transform that maps a 1-D signal into a 2-D time-frequency map, which describes how the spectral content of the data evolves with time. As such, TFD's are the natural tools for the analysis, synthesis, interpretation, and processing of nonstationary signals. Among the more well-known TFD's are the short-time Fourier transform (STFT) [1], the Gabor representation [17], and the Wavelet transform [33, 55].

Linearity is a desirable property of algorithms used in analyzing linear systems; however, quadratic time-frequency distributions have been proposed, analyzed, and interpreted as time-varying power spectra [13, 25]. The so-called Cohen class of shift-invariant distributions includes special cases such as the Spectrogram, Rihaczek, Page, Wigner-Ville distribution (WD) and Choi-Williams distributions [13, 25].

The WD has become quite popular because it possesses a number of useful properties, and has been used in the analysis of phase modulated signals, which are common in radar and sonar [3]. An important property of the WD is that every member of Cohen's class can be interpreted as a 2-D linearly filtered version of the WD [25]. A third-order WD was introduced in [18]; it was

generalized and its properties were studied in [14, 15, 60, 61, 62]. Higher-order WD's describe the evolution in time of the higher-order *moment spectra* of the signal.

Wigner Spectrum

The Wigner distribution (WD) was introduced in 1932 by Wigner in the context of quantum mechanics; its usefulness to problems in communication theory was discovered by Ville in 1948; consequently, it is often called the Wigner-Ville distribution. A series of papers by Classen and Mecklenbraüker [11] discussed the usefulness of the WD for time-frequency analysis of continuous and discrete-time signals, and was devoted to applications in digital signal processing; sampling issues are also discussed in [12]. Tutorial overviews of the WD and its relationships with other time-frequency distributions (TFD) may be found in [3, 13, 25].

In order to differentiate the second-order WD from the higher-order WD's (to be introduced later), we will refer to the conventional WD as the Wigner spectrum (WS).

In contrast to the STFT, which is resolution limited either in time or in frequency (dictated by the window function), and suffers from smearing and sidelobe leakage, the WS offers excellent resolution in both the frequency and time domains.

The Wigner cross spectrum (WCS) of two signals, $x(t)$, $y(t)$, is defined via,

$$W_{xy}(t, \omega) := \int x(t + \tau/2) y^*(t - \tau/2) e^{-j w \tau} \, d\tau \tag{1-170}$$

$$:= \frac{1}{2\pi} \int X(\omega + \xi/2) Y^*(\omega - \xi/2) e^{j \xi t} \, d\xi \tag{1-171}$$

where $X(\omega)$ and $x(t)$ constitute an FT pair. The auto WS is obtained when $x(t) = y(t)$, and is a bilinear function of the signal $x(t)$.

The *ambiguity function*, which is widely used in radar, is defined as the 2-D FT of the WS, and can also be expressed as

$$AF(\theta, \tau) = \int x^*\left(t - \frac{\tau}{2}\right) x\left(t + \frac{\tau}{2}\right) e^{j 2\pi \theta t} \, dt \tag{1-172}$$

Cohen's general class of TFD's is defined by

$$W_C(t, f) = \iint \phi(\theta, \tau) AF(\theta, \tau) e^{-j2\pi t\theta - j2\pi f\tau} \, d\tau \quad d\theta \tag{1-173}$$

where the kernel $\phi(\theta,\tau)$ specifies the particular distribution that is obtained.

Let $x(n) = y(n) + z(n)$; it follows that

$$W_{xx}(f, n) = W_{yy}(f, n) + W_{zz}(f, n) + W_{y,z}(f, n) + W_{z,y}(f, n) \tag{1-174}$$

Since the WS is a quadratic transform, the WS of the sum of two signals is not the sum of the individual WS's, but also has cross-terms. These cross-terms make it difficult to interpret the WS. These cross-terms may be suppressed by appropriate filtering, in the ambiguity function domain, that is, by appropriate choice of the kernel $\phi(\theta,\tau)$.

In order to reduce the effects of the cross-terms, Choi and Williams [10] proposed using the kernel

$$\phi(\theta, \tau) = \exp(-\theta^2\tau^2/\sigma^2) \tag{1-175}$$

where the parameter σ controls the amount of attenuation (the amplitude of the cross-terms is proportional to σ). Unfortunately, increased suppression of cross-terms invariably leads to smearing or loss of resolution of the auto terms in the time-frequency plane.

In practice, signals are sampled in time, and we compute FTs also on a sampled frequency grid. The discretization in time and frequency of the continuous-time WS leads to the nonaliasing requirement that the original signal be sampled at twice the Nyquist rate [12].

The discrete-time algorithm is given next [13, 25]. Let the instantaneous cross-correlation be defined by

$$r_{xy}(m, n) = x^*(n - m)y(n + m) \tag{1-176}$$

where n is identified with time and m with lag. The WCS is then defined by,

$$W_{xy}(f, n) = \sum_{m} r_{xy}(m, n) \exp(-j\pi f m) \qquad \textbf{(1-177)}$$

The WS is obtained when $x(n) = y(n)$. The original signal must be sampled at twice the Nyquist rate or faster, in order to avoid aliasing. In practice, the frequency variable f is also discretized, $f = k/K$, where K controls the frequency resolution. The WCS in (1-177) can also be implemented via two FFT algorithms, following the approach in [46]. Both approaches demand the same computational and storage complexity. This algorithm in (1-177) is implemented in routine `wig2`.

The ambiguity function, AF, can be computed as

$$AF(m, \theta) = \sum_{n} r_{xy}(m, n) \exp(j2\pi n\theta) \qquad \textbf{(1-178)}$$

The AF is multiplied by the Choi-Williams filter,

$$w(m, \theta) = \exp(-(m\theta/\sigma)^2) \qquad \textbf{(1-179)}$$

A 2-D FT (θ to n and m to f) yields the filtered WS. Cross-terms can usually be suppressed via this approach, but with a concomitant loss of resolution. In practice, θ is a discretized frequency grid. This algorithm is implemented in routine `wig2c`.

Examples

```
clf
load wigdat
subplot(221), wig2(s2,[],0);
subplot(222), wig2(s2);
subplot(223), wig2(s3);
subplot(224), wig2(s4);
```

You should see the display in Figure 1-20.

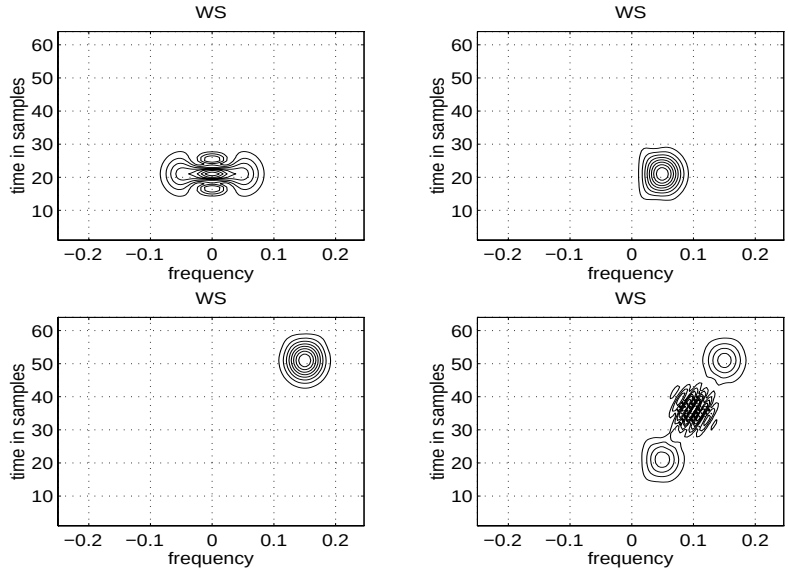


Figure 1-20 Wigner Spectra (wig2)

Signal s2 is a harmonic with a nominal frequency of 0.05 Hz, which has been multiplied by a Gaussian window, and is centered at $n = 20$. Note that the WS of the signal is concentrated around its nominal center frequency; note also the interaction between the energies at the positive and negative frequencies giving rise to the interference terms around D.C. By using the analytic version of the signal, the negative frequency terms are suppressed; this also suppresses the distortion terms around D.C.

Signal `s3` is a harmonic with a nominal frequency of 0.15 Hz, which has been multiplied by a Gaussian window, and is centered at $n = 50$. Note that the WS of the signal is concentrated around its nominal center frequency.

In the last panel, we compute the WS of the sum signal `s4=s2+s3`; note the cross-terms in the WS; some of these can be eliminated by using routine `wig2c`.

Examples

```
load wigdat
wig2c(s4);
```

Compare the display in Figure 1-21 with the (2,2) panel of Figure 1-20; note that the cross-terms have been suppressed.

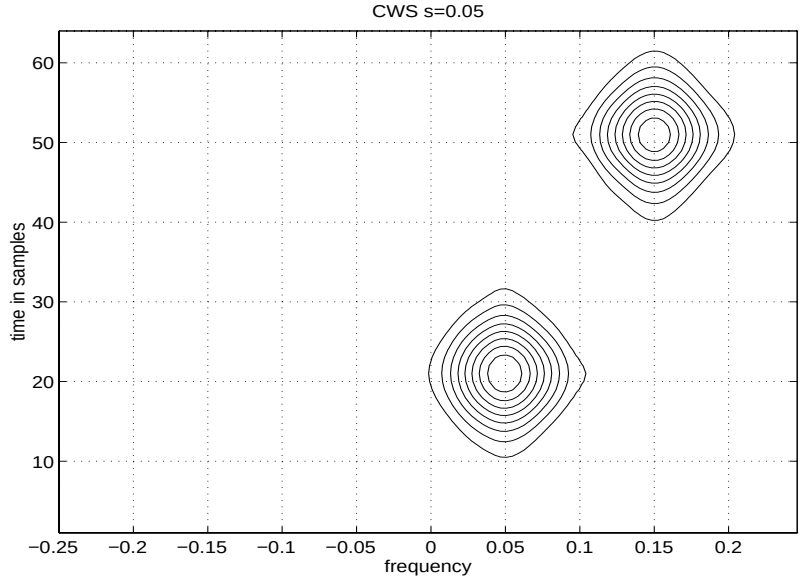


Figure 1-21 Wigner Spectrum With Choi-Williams Smoothing (wig2c)

Wigner Bispectrum

The notion of Wigner spectra can be generalized to Wigner bispectra (WB) and Wigner trispectra (WT) by considering the FTs of appropriately defined instantaneous triple and quadruple products.

Define the instantaneous triple-product

$$r_3(t, \tau_1, \tau_2) = x^*(t - \alpha\tau_1 - \alpha\tau_2)x(t + \beta\tau_1 - \alpha\tau_2)x(t - \alpha\tau_1 + \beta\tau_2) \quad \textbf{(1-180)}$$

where $\alpha = 1/3$ and $\beta = 2/3$. The WB is then given by

$$W(n, f_1, f_2) = \iint d\tau_1 d\tau_2 e^{-j2\pi(f_1\tau_1 + f_2\tau_2)} r_3(t, \tau_1, \tau_2) \quad \textbf{(1-181)}$$

This algorithm is implemented in routine `wig3`, where the slice $f_1 = f_2$ is computed. Note that the original signal should be sampled at twice the Nyquist rate, in order to avoid aliasing. It is shown in that the frequency axes are scaled by the factor of 2/3, which can be easily fixed.

As in the case of the WS, the WB also suffers from the problems of cross-terms; one can attempt to suppress these terms by appropriate filtering. Define the smoothing kernel,

$$\Phi(\theta, \tau_1, \tau_2) = \exp(-\theta^2(\tau_1^2 + \tau_2^2)/\sigma) \quad \textbf{(1-182)}$$

The filtered WB is then given by

$$W(n, f_1, f_2) = \iiiint d\theta\tau_1 d\tau_2 du e^{-j2\pi(f_1\tau_1 + f_2\tau_2)} e^{-j2\pi t\theta} e^{j2\pi u\theta} r_3(t, \tau_1, \tau_2)\Phi(\theta, \tau_1, \tau_2)$$

This algorithm is implemented in routine `wig3c`, where the slice $f_1 = f_2$ is computed. It should also be noted that the application of the Choi-Williams filter to the WB does not guarantee preservation of the auto-terms; consequently, routine `wig3c` should be used with great caution.

Examples

```
clf
load wigdat
subplot(221), wig3(s2,[],0);
subplot(222), wig3(s2);
subplot(223), wig3(s3);
subplot(224), wig3(s4);
```

You should see the display in Figure 1-22.

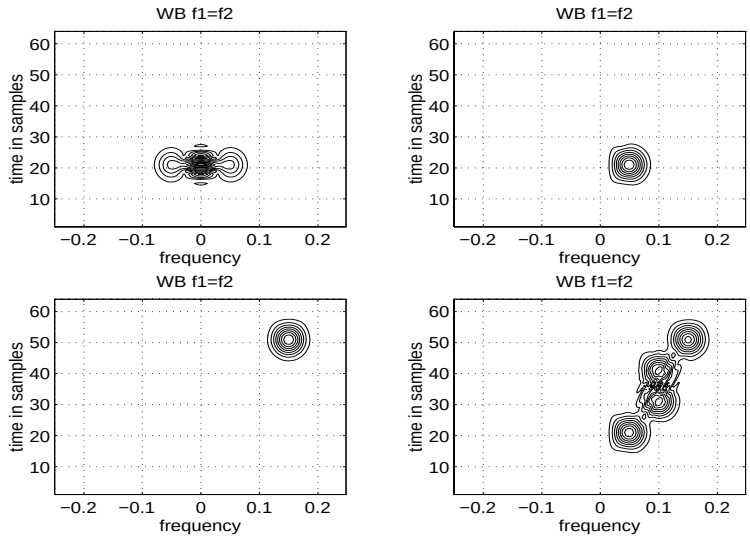


Figure 1-22 Wigner bispectra (wig3)

Signal s2 is a harmonic with a nominal frequency of 0.05 Hz, which has been multiplied by a Gaussian window, and is centered at $n = 20$. Note that the WB of the signal is concentrated around its nominal center frequency; note also the interaction between the energies at the positive and negative frequencies giving rise to the interference terms around D.C. By using the analytic version of the signal, the negative frequency terms are suppressed; this also suppresses the distortion terms around D.C.

Signal `s3` is a harmonic with a nominal frequency of 0.15 Hz, which has been multiplied by a Gaussian window, and is centered at $n = 50$. Note that the WB of the signal is concentrated around its nominal center frequency.

In the last panel, we compute the WB of the sum signal `s4=s2+s3`; note the cross-terms in the WS; some of these can be eliminated by using routine `wig3c`.

A direct consequence of the definition of the WB is that the frequency axes are scaled by the factor 2/3 [4]–[5]. In this routine, this scaling has been undone so that the peaks appear at the *expected* frequencies.

Examples

```
load wigdat
wig3c(s4,256,0.2,1)
```

Compare the display in Figure 1-23 with the (2,2) panel in Figure 1-20 and with Figure 1-21; notice the suppression of cross-terms.

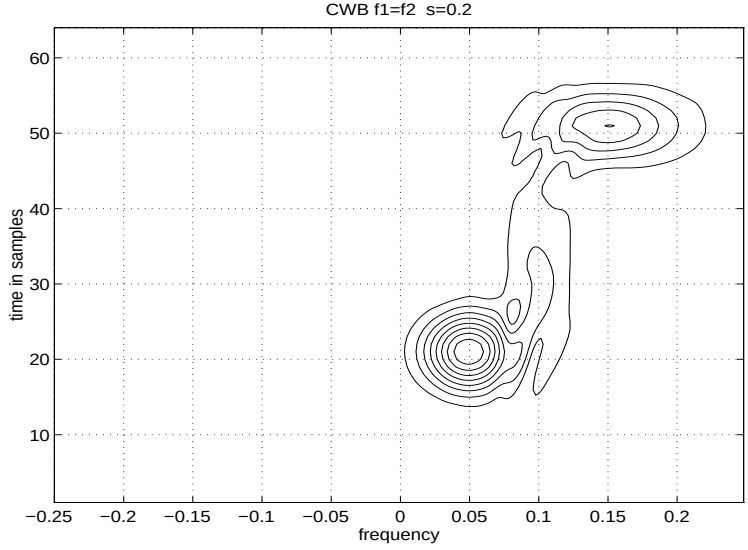


Figure 1-23 Wigner bispectrum With Choi-Williams Smoothing (wig3c)

Wigner Trispectrum

Define the instantaneous fourth-order product,

$$r_4(t, \tau_1, \tau_2, \tau_3) = x^*(t-\tau)x(t-\tau+\tau_1)x(t-\tau+\tau_2)x^*(t-\tau+\tau_3) \quad \textbf{(1-183)}$$

where $\tau := (\tau_1 + \tau_2 + \tau_3)/4$.

The WT is then given by

$$W(n, f_1, f_2, f_3) = \iiint d\tau_1 d\tau_2 d\tau_3 e^{-j2\pi(f_1\tau_1 + f_2\tau_2 - f_3\tau_3)} r_4(t, \tau_1, \tau_2, \tau_3) \quad \textbf{(1-184)}$$

Notice that two terms are conjugated; also note the sign of the $f_3\tau_3$ term. This algorithm is implemented in routine `wig4` where the slice $f_1 = f_2 = -f_3 = f$ is computed, that is, we compute

$$W_s(n, f) = \iiint d\tau_1 d\tau_2 d\tau_3 e^{-j2\pi f(\tau_1 + \tau_2 + \tau_3)} r_4(t, \tau_1, \tau_2, \tau_3) \quad \textbf{(1-185)}$$

Note that the original signal should be sampled at twice the Nyquist rate, in order to avoid aliasing. Also note that the frequency axes are scaled by the factor of 1/2, which can be easily fixed.

The WT also suffers from the presence cross-terms. As in the case of WS, we can attempt to suppress the cross terms by appropriate filtering.

Define the smoothing kernel,

$$\Phi(\theta, \tau_1, \tau_2, \tau_3) = \exp(-\theta^2(\tau_1^2 + \tau_2^2 + \tau_3^2)/\sigma) \quad \textbf{(1-186)}$$

The filtered WT is then given by

$$W(n, f_1, f_2, f_3) = \iiiint d\theta d\tau_1 d\tau_2 d\tau_3 du\, e^{-j2\pi(f_1\tau_1 + f_2\tau_2 + f_3\tau_3)}$$

This algorithm is implemented in routine `wig4c`, where the slice $f_1 = f_2 = -f_3$ is computed; the resulting sliced WT is real valued. As in the case of the WS, cross-terms can usually be suppressed by this approach, but with a concomitant loss of resolution. In contrast to the WB, the filtering does not unduly distort the auto terms [14, 15].

Examples

```
clf, load wigdat
subplot(221), wig4(s2,[],0);
subplot(222), wig4(s2);
subplot(223), wig4(s3);
subplot(224), wig4(s4);
```

You should see the display in Figure 1-24. Signal `s2` is a harmonic with a nominal frequency of 0.05 Hz, which has been multiplied by a Gaussian window, and is centered at $n = 20$. Note that the WT of the signal is concentrated around its nominal center frequency; note also the interaction between the energies at the positive and negative frequencies giving rise to the interference terms around D.C. By using the analytic version of the signal, the negative frequency terms are suppressed; this also suppresses the distortion terms around D.C.

Signal `s3` is a harmonic with a nominal frequency of 0.15 Hz, which has been multiplied by a Gaussian window, and is centered at $n = 50$. Note that the WT of the signal is concentrated around its nominal center frequency.

In the last panel, we compute the WS of the sum signal `s4=s2+s3`; note the cross-terms in the WT; some of these can be eliminated by using routine `wig4c`.

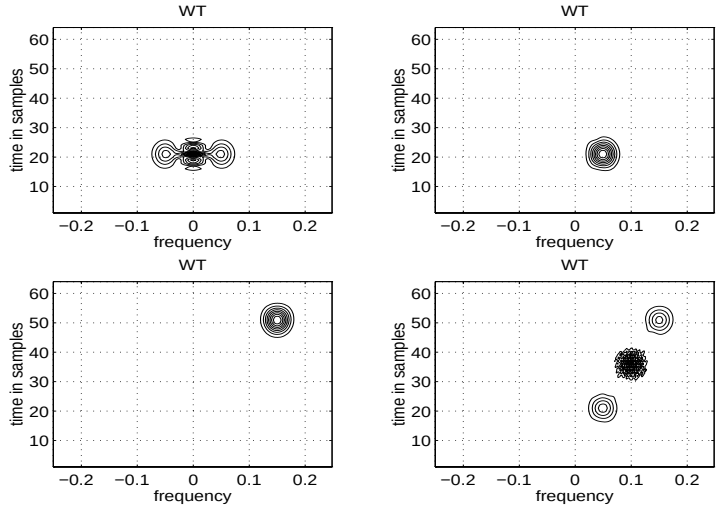


Figure 1-24 Wigner trispectra (wig4)

Examples

```
load wigdat
wig4c(s4);
```

Compare the display in Figure 1-25 with the (2,2) panel of Figure 1-24, and Figure 1-21 and Figure 1-23; note that the cross-terms in Figure 1-24 have been suppressed.

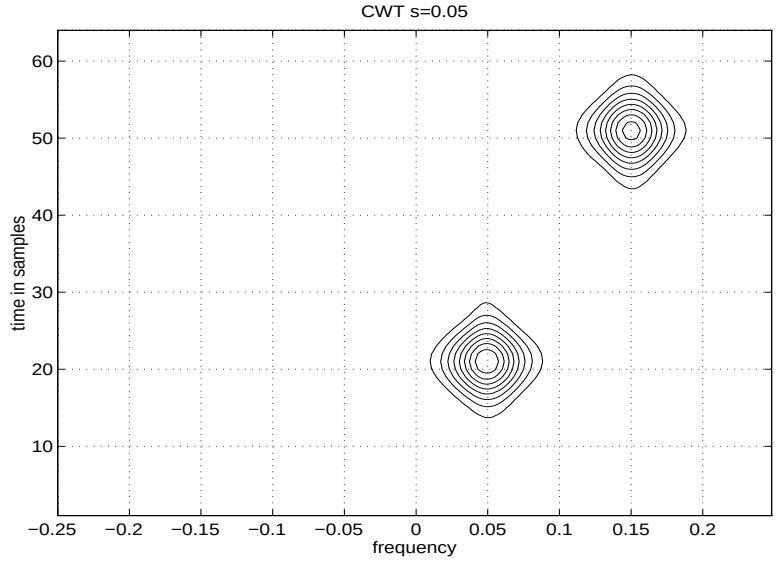


Figure 1-25 Wigner Trispectrum With Choi-Williams Filtering (wig4c)

Summary

The Higher-Order Spectral Analysis Toolbox offers routines to compute the WS, and slices of the WB and the WT, and are implemented in `wig2`, `wig3`, and `wig4`. Routines `wig2c`, `wig3c`, and `wig4c` implement the so-called Choi-Williams filter, which can help suppress the cross-terms in the WS, WB, and WT. It should be noted that this filtering may, in fact, destroy signal terms in the WB (but not in WS or WT). Wigner spectra of different orders offer different perspectives on the data, and are useful for exploratory data analysis.

Time-Delay Estimation

The time-delay estimation problem occurs in various applications, for example, in the determination of range and bearing in radar and sonar. It also has esoteric applications, such as the measurement of the temperature of a molten alloy by measuring the passage time of a signal. Other applications include analysis of EEG data.

The basic model is as follows: two sensors record delayed replicas of a signal, in the presence of noise:

$$x(t) = s(t) + w_x(t) \tag{1-187}$$

$$y(t) = As(t - D) + w_x(t) \tag{1-188}$$

where D is the delay of the signal at the y-sensor relative to the signal at the x-sensor, A is the relative amplitude gain, and $w_x(t)$ and $w_y(t)$ are sensor noises. Given $x(t)$, $y(t)$, t=0 . . ., $N - 1$, we want to estimate the delay D.

The basic idea is to shift the signal $y(t)$ and compare the shifted waveform with $x(t)$; the best match occurs when the shift equals the delay D. This notion is made more precise by using the cross-correlation between the two signals. We assume that $s(t)$ is a stationary process, and that the noises are zero mean.

A Cross-Correlation Based Method

The cross-correlation between the two signals $x(t)$ and $y(t)$ is given by,

$$R_{xy}(\tau) = AR_{ss}(\tau - D) + R_{w_x, w_y}(\tau) \tag{1-189}$$

where $R_{w_x, w_y}(\tau)$ is the cross correlation between the two noise processes, and $R_{ss}(\tau)$ is the autocorrelation of the signal. If the noises are uncorrelated, $R_{xy}(\tau)$ will have a peak at $\tau = D$, the unknown delay. In practice, due to effects of finite length estimates, and due to the presence of noise, the cross-correlation estimate may not have a sharp peak.

The data may be prefiltered in order to sharpen the peak; equivalently, we can multiply the estimated cross-correlation by a window function. Different choices of the window function lead to different estimates. The most popular

window function is the *maximum-likelihood* window of Hannan and Thompson, which is described below.

Let $S_{xy}(f)$ denote the cross-spectrum between the two signals, x and y; and let $S_{xx}(f)$ and $S_{yy}(f)$ denote the autospectra of x and y. The squared coherence function is defined by

$$C_{xy}(f) = \frac{|S_{xy}(f)|^2}{S_{xx}(f)S_{yy}(f)}.$$

The optimal-maximum-likelihood window is then

$$W(f) = \frac{1}{|S_{xy}(f)|}\frac{C_{xy}(f)}{(1 - C_{xy}(f))},$$

and the windowed cross-correlation, $R_{xy}(m)$, is the IFT of $W(f)S_{xy}(f)$.

Estimates of the auto- and cross- spectra and the coherence can be obtained via the MATLAB routine `spectrum`; the segment length must be at least twice the expected maximum delay. Since good estimates of the spectra demand a large number of segments, it is critical that the lengths of the time-series, x and y, be much larger than the expected maximum delay.

An initial estimate, d, of the delay D is given by the location of the peak of $R(m)$. A three-point interpolation may be used to improve the delay estimate [39]

$$\hat{D} = \frac{2D - 1}{2} - \frac{R(d) - R(d-1)}{R(d+1) - 2*R(d) + R(d-1)}.$$

The optimal-maximum-likelihood window estimator is implemented in routine `tder`.

Examples

```
load tde1
delay=tder(s1,s2,30,64,0,64);
Estimated delay= 15.9987
```

You should see the display on Figure 1-26 and the Estimated delay (Estimated=15.997). The two signals are corrupted by spatially correlated noise; the noise correlation shows a strong peak at a delay of 5 samples; the signal delay is 16 samples.

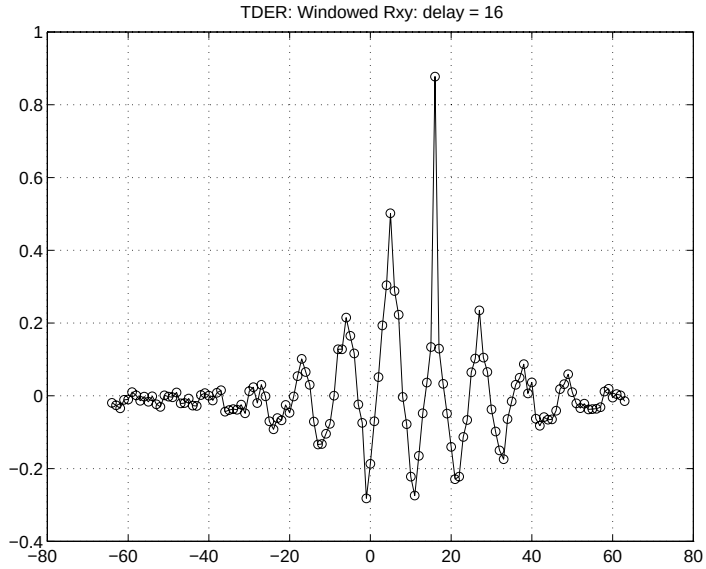


Figure 1-26 Time-Delay Estimated Using Cross-Correlation (tder)

A Cross-Cumulant Based Method

The cross-correlation based method assumes that the sensor noises are uncorrelated. If the noise processes are correlated, it may not be possible to detect the peak of $R_{ss}(\tau)$ since it may be masked by $R_{w_x, w_y}(\tau)$ (see (1-189)). If the signals are non-Gaussian, and the noise processes are Gaussian, we can use third-order cumulants, even if the noise processes are correlated.

Let P be the maximum expected delay, and assume that the delay D is an integer. Then, [39, 40]

$$y(n) = \sum_{i=-P}^{P} a(i)x(n-i) + w(n) \tag{1-190}$$

where $a(n) = 0$, $n \neq D$, and $a(D) = 1$. Consider the third-order cumulants,

$$C_{yxx}(\tau, \rho) := E\{y^*(n)x(n+\tau)x(n+\rho)\} \tag{1-191}$$

$$C_{xxx}(\tau, \rho) := E\{x^*(n)x(n+\tau)x(n+\rho)\} \tag{1-192}$$

Substituting (1-190) into (1-191), we obtain

$$C_{yxx}(\tau, \rho) = \sum_{i=-P}^{P} a(i)C_{xxx}(\tau + i, \rho + i) \tag{1-193}$$

Using this equation for various values of ρ and τ, we get a system of linear equations in the $a(i)$'s, namely,

$$\mathbf{C}_{xxx}\mathbf{a} = \mathbf{c}_{yxx}.$$

The estimated delay is the index n which maximizes $|a(n)|$. A low rank approximation of the cumulant matrix $\mathbf{C}_{xxx}$ may be used to improve the robustness to noise. This algorithm is implemented in routine `tde`.

Examples

```
load tde1
[delay,avec]=tde(s1,s2,30,128);
Estimated delay= 16
```

You should see the display in Figure 1-27, and the Estimated delay (Estimated delay=16). The two signals are corrupted by spatially correlated noise; the cross-correlation between the noises at the two sensors has a strong peak at a delay of 5 samples; the signal delay is 16 samples.

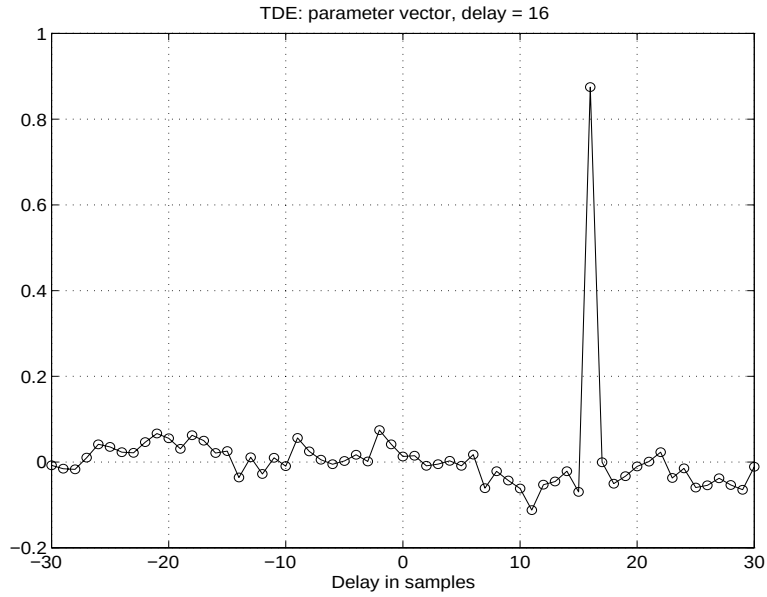


Figure 1-27 Time-Delay Estimated Using Cross-Cumulants (tde)

A Hologram Based Method

The algorithm described in the previous section can also be implemented in the frequency domain. Define auto- and cross-bispectra,

$$B_{xxx}(f_1, f_2) = E\{X(f_1)X(f_2)X^*(f_1 + f_2)\} = S_{xxx}(f_1, f_2)$$

$$B_{xyz}(f_1, f_2) = E\{X(f_1)X(f_2)X^*(f_1 + f_2)\} = S_{xxx}(f_1, f_2)e^{-j2\pi Df_2}$$

The third-order *hologram*, $h(\tau)$, is then defined by [39, 40]

$$h(\tau) = \int df_1 \int df_2 \exp(j2\pi f_2 \tau) \frac{B_{xyx}(f_1, f_2)}{B_{xxx}(f_1, f_2)} \tag{1-194}$$

$$= \int df_1 \int df_2 \exp(j2\pi f_2 \tau) e^{-j2\pi D f_2} \tag{1-195}$$

$$= \delta(\tau - D) \tag{1-196}$$

The hologram should display a strong peak at the location of the true delay. Since third-order statistics are used, the method is insensitive (in theory) to both spatially and temporally colored noise which is symmetrically distributed (e.g., Gaussian). In practice, the estimated hologram, $h(\tau)$, will not be an impulse; hence, we estimate D as the index τ, which maximizes $|h(\tau)|$. This algorithm is implemented in routine `tdeb`.

Examples

```
load tde1
delay=tdeb(s1,s2,30);
Delay estimated by TDEB is 16
```

You should see the display on Figure 1-28, and the Estimated delay (Estimated delay=16). The two signals are corrupted by spatially correlated noise; the cross-correlation between the noises at the two sensors has a strong peak at a delay of 5 samples; the signal delay is 16 samples.

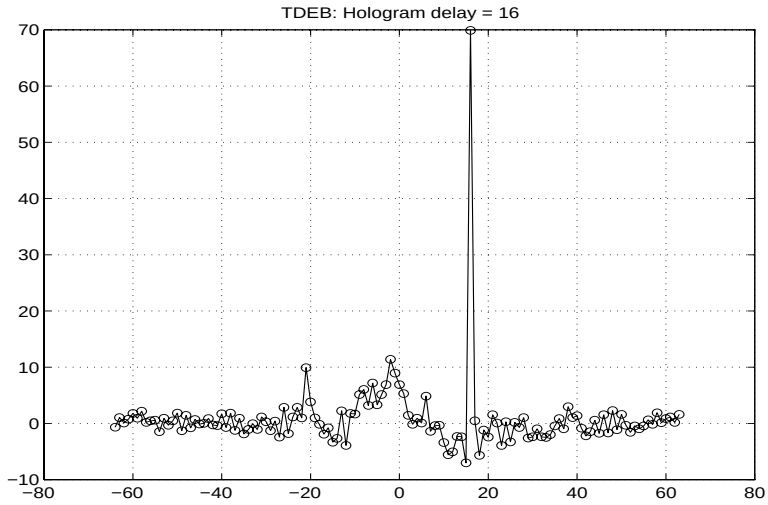


Figure 1-28 Time-Delay Estimated Using Cross-Bispectrum (tdeb)

Summary

The basic idea in time-delay estimation is to locate the peak in the cross-correlation between two signals; routine `tder` implements the *Maximum-likelihood* window of Hannan and Thompson, and is useful if the noises at the two sensors are not spatially correlated.

Routines `tde` and `tder` use third-order cross-cumulants and cross-bispectra respectively; hence, the sensor noises may be spatially correlated, provided they are symmetric (cross-bispectrum of the noises is zero). Routine `tdegen` can be used to generate synthetics for the TDE problem.

Case Studies

In this section, we will use the Higher-Order Spectral Analysis Toolbox to process some real data; in doing so, we will learn some of the pitfalls and tricks of the trade.

Sunspot Data

Schuster was, perhaps, the first to use the periodogram to analyze sunspot data in the nineteenth century; subsequently, these data have been virtually adopted as a standard by statisticians and signal analysts. These data are known to have an approximate 11-year cycle. Sunspot data for the years 1700-1987 are available in the file `sunspot.dat`, which is included in the standard distribution package.

Figure 1-29 shows the data and the histogram; the latter is indicative of an exponential distribution. We used `cumest` to obtain the estimates of some summary statistics, which are shown in the third column of Table 1-1; the last column is discussed later.

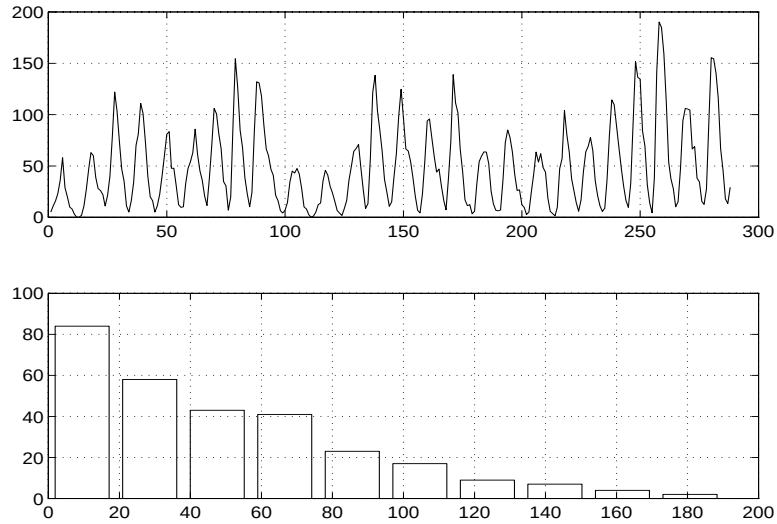


Figure 1-29 Sunspot Data

Table 1-1: Summary Statistics for Sunspot Data

		Original	Differenced
Mean	$\mu = E\{x(t)\}$	48.434	0.085
Variance	$\sigma^2 = E\{(x(t) - \mu)^2\}$	1548.781	547.773
Skewness	$E\{(x(t) - \mu)^3\}/\sigma^3$	1.038	0.904
Kurtosis	$E\{(x(t) - \mu)^4\}/\sigma^4 - 3$	0.657	1.502

The data in Figure 1-29 display an apparent periodicity; we can use `harmest` to verify this. The (1,1) panel of Figure 1-30 displays the singular values of the covariance matrix; the plot indicates that only three singular values are significant ($p = 3$); hence, we used an order of 3. The corresponding power spectral estimates are shown in the remaining panels of Figure 1-30: All of the estimates have a strong peak at 0 Hz and at approximately ±0.1 Hz. We can use `pickpeak` to accurately locate the peaks in the power spectra; we can also use **`roots`** to estimate the locations of the roots of the AR polynomial estimated by the AR method; doing so, we obtain an estimate of 10.64 years for the period.

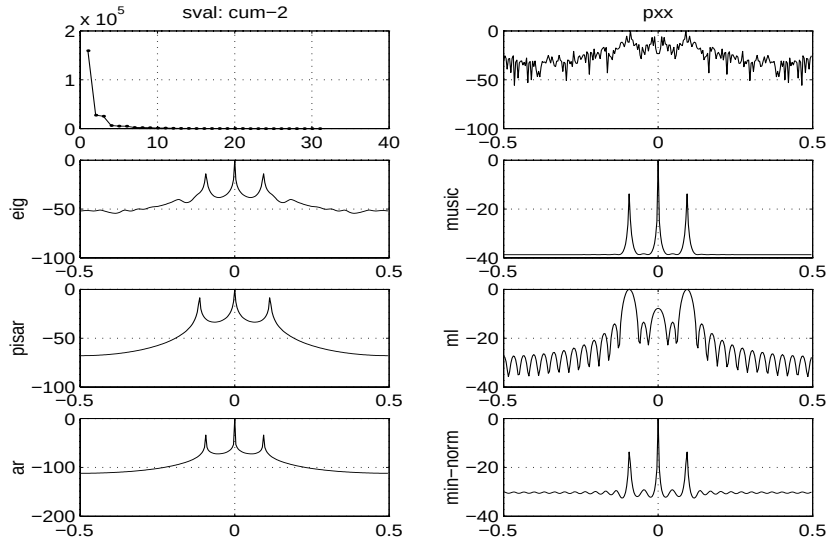


Figure 1-30 Sunspot Data: Power Spectra

We used `glstat` to test for Gaussianity and linearity; the test results are:
Test statistic for Gaussianity is 357.4639 with df = 60, Pfa = 0
Linearity test: R (estimated) = 14.8592, lambda = 11.0332, R (theory) = 9.1222, N = 16.

The probability of false alarm (Pfa) in rejecting the Gaussian hypothesis is 0, that is, we are virtually certain that the data have nonzero bispectrum; thus, the test indicates that the data are non-Gaussian as expected (recall that the histogram shows that the univariate pdf is non-Gaussian, and that the skewness is around unity). The estimated and theoretical values of R, the interquartile range of the estimated bicoherence values, are not very different from one another; hence, the tests do not show any strong evidence of nonlinearity.

The bispectrum of the data was estimated via `bispeci`; we used 25 lags and the default window. The resulting estimate is shown in Figure 1-31. The bispectrum shows sharp peaks at around (0,0.1) and (0.1,0.1) (and symmetric locations) and is indicative of possible quadratic coupling. The peak at (0,0.1) is most probably an artifact due to the fact that the data are all positive valued.

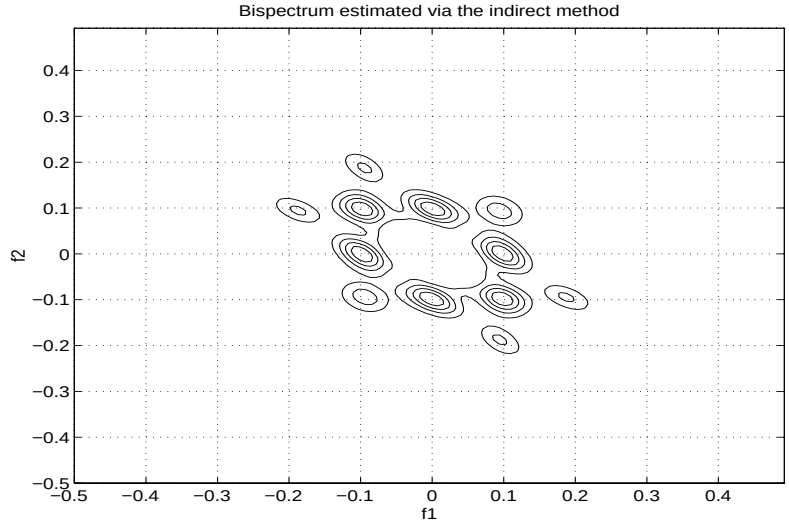


Figure 1-31 Sunspot Data: Bispectrum

Differencing the data is a standard trick in exploratory data analysis; we can use MATLAB's `diff` function to compute the first-order differences. The differenced data and the corresponding histogram are shown in Figure 1-32. Summary statistics are shown in the last column of Table 1-1.

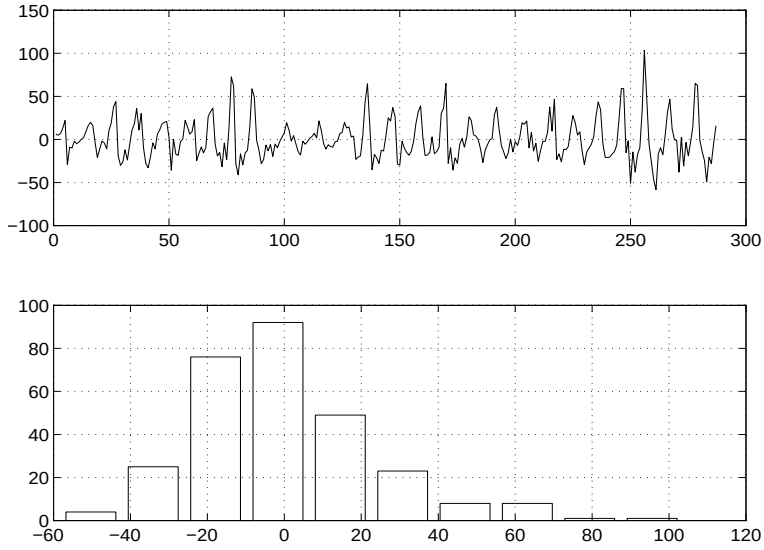


Figure 1-32 Sunspot Data: First Difference

The singular value plot displayed by `harmest` (see the (1,1) panel in Figure 1-33) indicates either four or six significant singular values; using p=6, we obtain the power spectra shown in the remaining panels of Figure 1-33. Again, using roots, we estimate the periods to be 5.47 and 10.50 years (AR method).

We used `glstat` to test for Gaussianity and linearity; the test results are:
Test statistic for Gaussianity is 250.1965 with df = 70, Pfa = 0
Linearity test: R (estimated) = 13.5335, lambda = 6.4449, R (theory) = 7.0433, N = 16.

The test indicates that the data are non-Gaussian as expected (recall that the skewness is around unity).

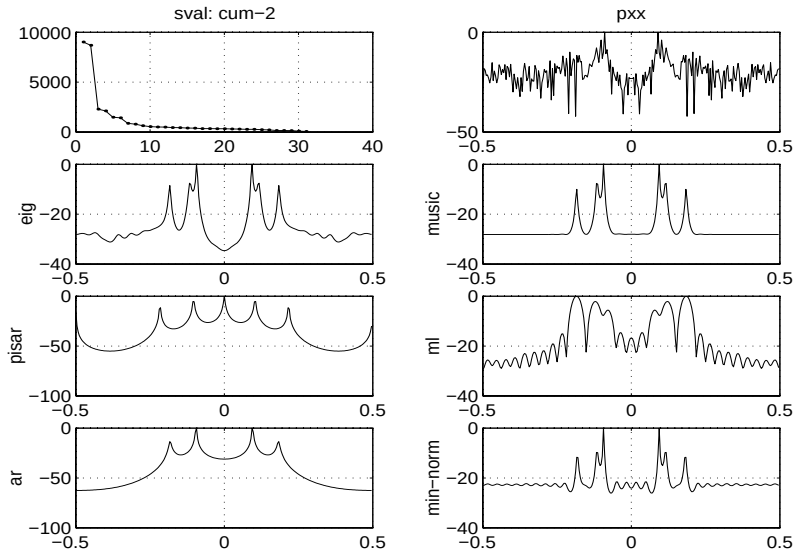


Figure 1-33 Differenced Sunspot Data: Power Spectra

The bispectrum of the differenced data in Figure 1-34 has sharp peaks around (.094, .094), and indicates coupling between the periods at 5.3 years and 10.6 years.

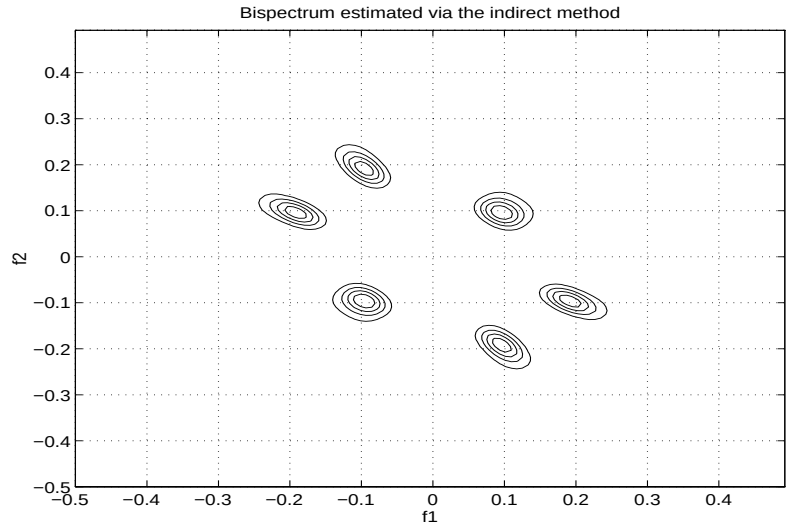


Figure 1-34 Differenced Sunspot Data: Bispectrum

In this example, differencing the data helped to clarify the estimates of the power spectrum as well as the bispectrum. Our tests indicate that the data are non-Gaussian, and show evidence of a fundamental period of around 10.6 years as well as a harmonic at around 5.3 years; this may be indicative of quadratic nonlinearities.

The various figures were generated via

```
load sunspot.dat; sp=sunspot(:,2); eda(sp); eda(diff(sp));
```

Function eda is listed in Table 1-3.

Canadian Lynx Data

Examples

This time series consists of the annual number of Canadian lynx trapped in the Mackenzie River district of Northwest Canada for the years 1821–1934; listings of the data may be found in [56] and [52], where this time series is discussed in detail, and additional references are also given.

We used eda to generate Figure 1-35 through Figure 1-40. The histogram in Figure 1-35 is indicative of an exponential distribution, and the data show apparent periodicity. Summary statistics are shown in Table 1-2; the last column corresponds to first differences.

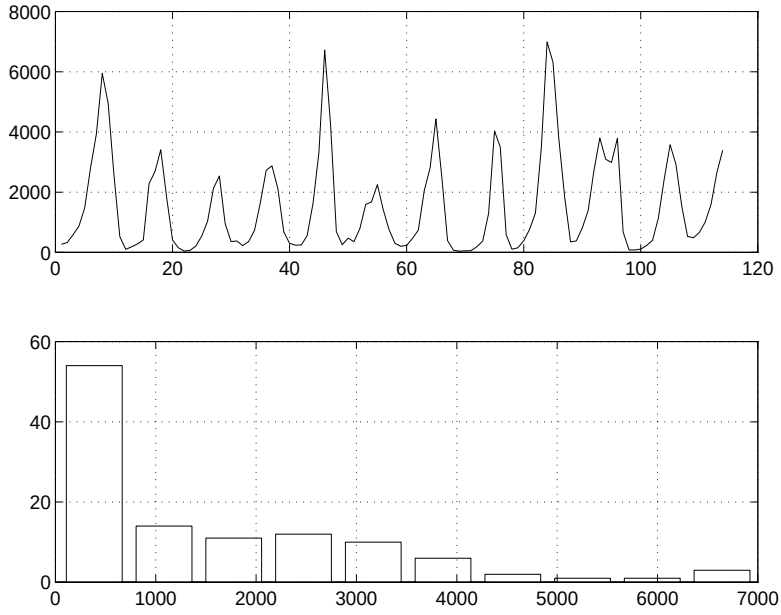


Figure 1-35 Lynx Data

Table 1-2: Summary Statistics for Lynx Data

		Original	Differenced
Mean	$\mu = E\{x(t)\}$	1537.89	27.673
Variance	$\sigma^2 = E\{(x(t) - \mu)^2\}$	2.439 e+06	1.409 e+06
Skewness	$E\{(x(t) - \mu)^3\}/\sigma^3$	1.345	-0.282
Kurtosis	$E\{(x(t) - \mu)^4\}/\sigma^4 - 3$	1.463	1.365

Routine `harmest` indicates an order of three (see panel (1,1) of Figure 1-36), and the power spectra are displayed in Figure 1-36. We see a peak at DC, and two other peaks corresponding to a cycle of 9.63 years.

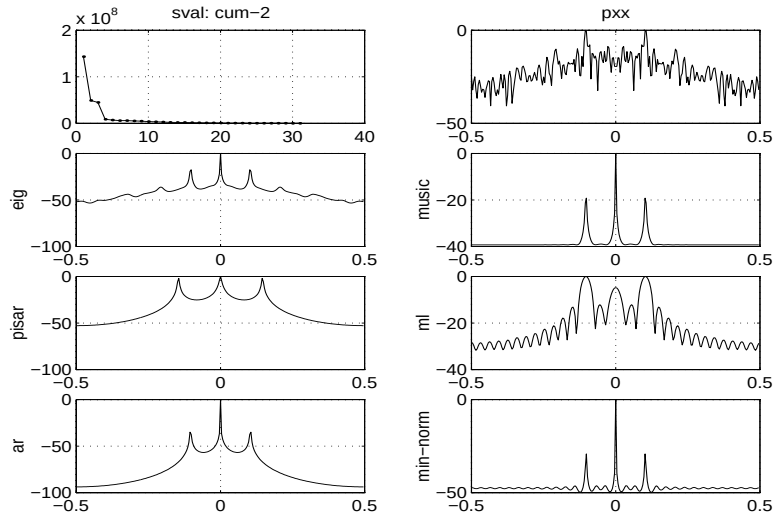


Figure 1-36 Lynx Data: Power Spectra

We used `glstat` to test for Gaussianity and linearity; the test results are:
Test statistic for Gaussianity is 196.752 with df = 28, Pfa = 0
Linearity test: R (estimated) = 6.8468, lambda = 11.299, R (theory) = 9.2282, N= 5.

The test indicates that the data are non-Gaussian as expected (recall that the skewness is around unity, and that the histogram is indicative of an exponential distribution). The time-series length is only 114 samples, and `glstat` bases its estimate of the interquartile range on only 5 samples; hence, the test for linearity is not conclusive.

The bispectrum of the data was estimated via `bispeci`; we used 25 lags and the default window. The resulting estimate is shown in Figure 1-37. The bispectrum shows sharp peaks at around (0.1,0.1) (and symmetric locations) and is indicative of possible quadratic frequency coupling. The data appear to be band-pass in nature.

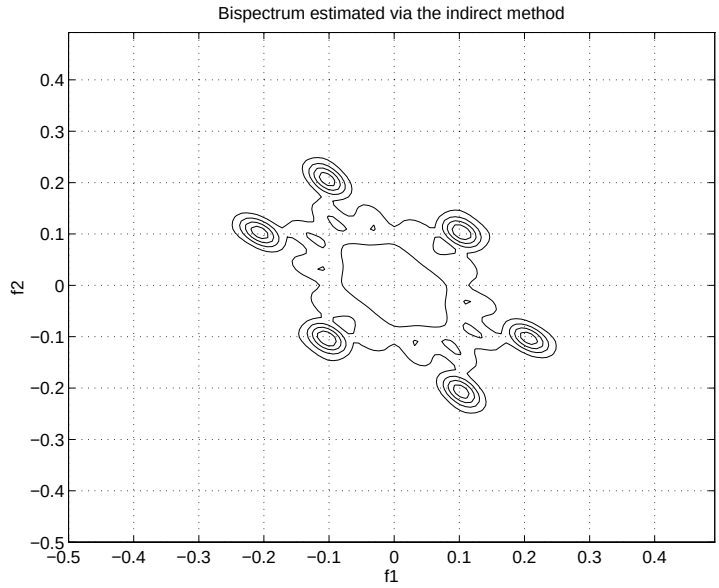


Figure 1-37 Lynx Data: bispectrum

As with the sun-spot data, we repeated the exercise with the first differences; corresponding plots are shown in Figure 1-38 through Figure 1-40. Notice that the histogram appears to be somewhat symmetrical, but the skewness is significantly nonzero; the correlation matrix indicates two harmonics (order 4) as evidenced by the singular value plot shown in the (1,1) panel of Figure 1-38; the estimated cycles are 4.8 years and 9.6 years. The bispectrum confirms the coupling between these two periods.

In this example, we used `harmest` and `bispeci` to estimate the periods of the harmonics, and to detect quadratic coupling; differencing the data helped to clarify the estimates of the power spectrum as well as the bispectrum. Our tests confirm that the data are non-Gaussian, and show evidence of a fundamental period of around 9.6 years as well as a harmonic at around 4.8 years; this may be indicative of quadratic nonlinearities.

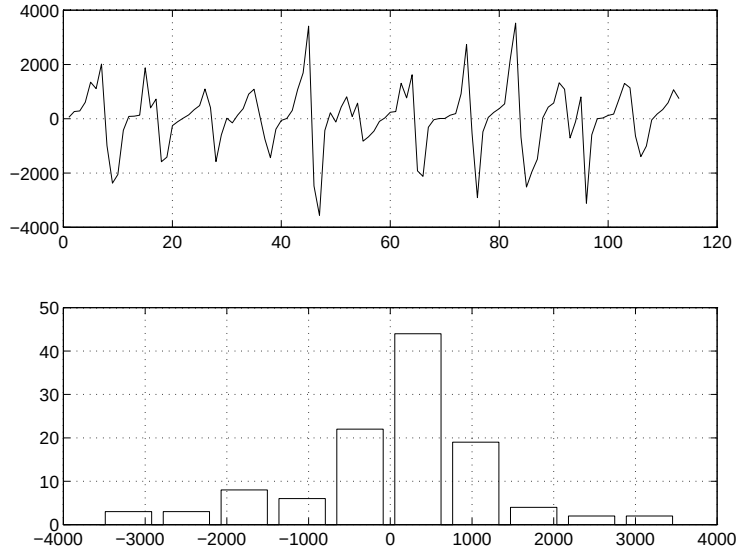


Figure 1-38 Lynx Data: Differenced

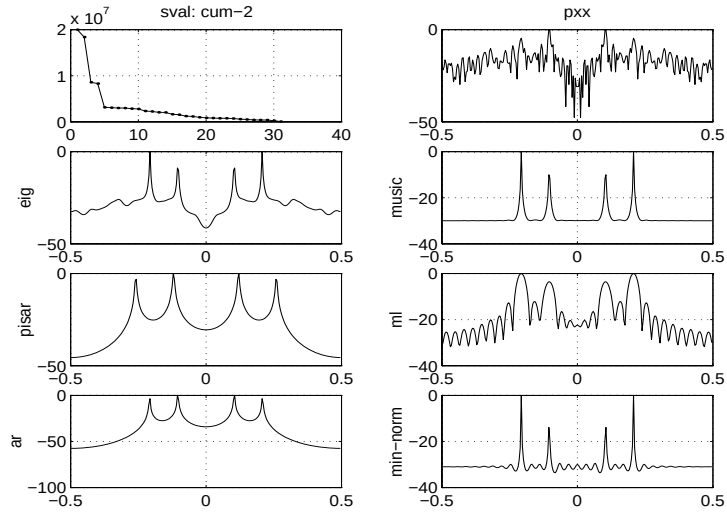


Figure 1-39 Lynx Data, Differenced: Power Spectra

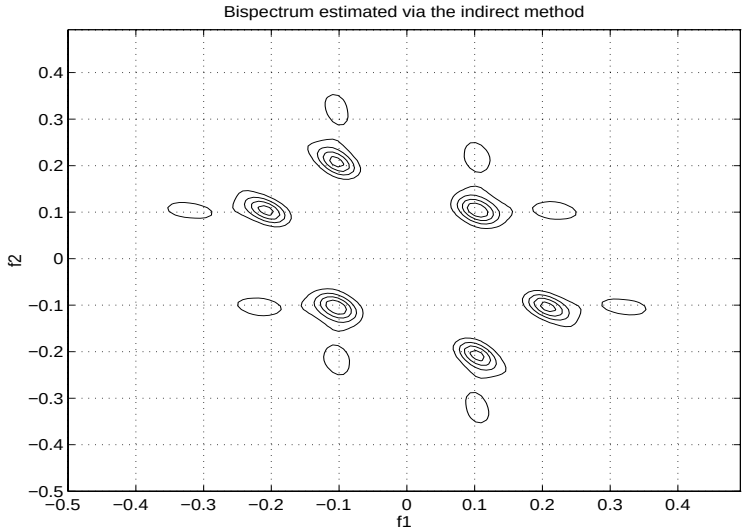


Figure 1-40 Lynx Data, Differenced: bispectrum

Table 1-3: Listing of eda.m

```
function  eda(sp)
%EDA   Exploratory Data Analysis
%   eda(sp)   :  sp is the time-series data
%
& Author: A. Swami, Jan 1995

disp('-------- 1 – data and histogram')
figure(1)
subplot(211), plot(1:length(sp),sp),grid,
subplot(212), hist(sp), grid

disp('-------- 2 – Summary stats')
c1 = mean(sp);
c2 = cumest(sp,2);
c3 = cumest(sp,3) / c2^(3/2);
c4 = cumest(sp,4) / c2^2;

fprintf(' Mean                    %g\n',c1);
fprintf(' Variance                %g\n',c2);
fprintf(' Skewness (normalized)   %g\n', c3);
fprintf(' Kurtosis (normalized)   %g\n', c4);

disp('-------- 3 – power spectra and harmonic models')
figure(2)
[p2,a2,b2] = harmest(sp, 30, 0, 'u', 256, 2);
r = cplxpair(roots(a2));
disp('Estimated cycles')
a = angle(r);
yest = a*0;
ind = find(a ~= 0) ;
yest(ind) = (2*pi) * ones(size(ind)) ./ angle(r(ind))  ;
yest

disp('-------- 4 – gaussianity and linearity tests')
glstat(sp, .51, length(sp) ) ;

disp('-------- 5 – the bispectrum')
figure(3)
[Bspec, w] = bispeci(sp, 25);     % 25 lags

%pcolor(w,w,abs(Bspec)), shading('interp')  % nice, slow
```

A Classification Example

The data consist of two underwater acoustic signals; the sampling rate is fairly high and we have lots of data (131,072 samples) The primary interest here is to classify the two signals.

Means and standard deviations were estimated over 262 nonoverlapped segments, each of length 500 samples, and are shown in Figure 1-41; the data appear to be highly nonstationary. Because of the nonstationarity, overall estimates of the power spectrum or the bispectrum, or tests of Gaussianity and or linearity will not be meaningful.

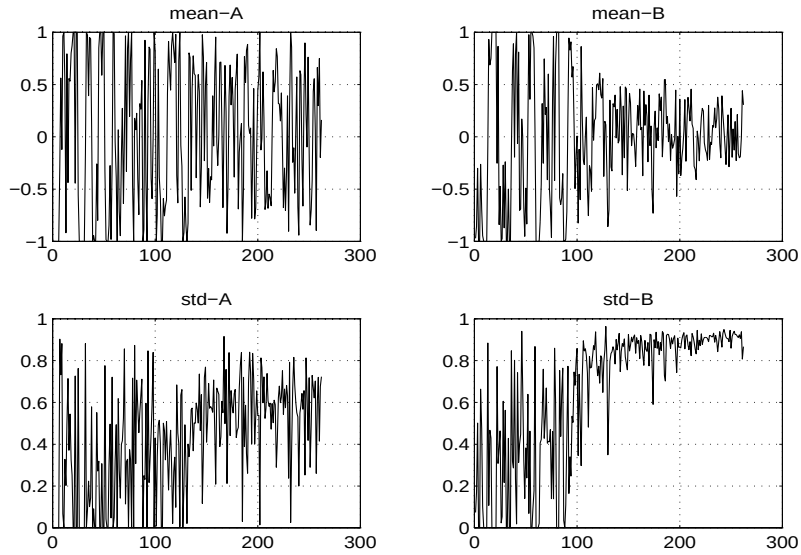


Figure 1-41 Acoustic Data: Means and Standard Deviations

We segmented the data into lengths of 1024 samples; power spectra, estimated from each segment, indicate that the data are essentially low-pass; we expect that a second-order AR model may be adequate to classify the data. We used

`arrcest`, with `p=2`, `q=0`, `norder=2`, `maxlag=24` to fit AR(2) models to successive segments of the data.

```
nsamp = 1024; ind=[1:nsamp]';
for k=1:length(x)/nsamp
    arx(:,k) = arrcest(x(ind),2,0,2,24);
    ary(:,k) = arrcest(y(ind),2,0,2,24);
    ind = ind + nsamp;
end
plot(arx(:,2),arx(:,3),'x', ary(:,2),ary(:,3),'o'),grid
```

In Figure 1-42 the AR coefficient $a(2)$ is plotted against the coefficient $a(1)$; circles and crosses correspond to the two time series; note that the data appear to be well-separated in the AR coefficient domain. The relationship between $a(2)$ and $a(1)$ appears to be linear; we used `tls` to estimate the slope and intercept, from which we derived the line separating the two classes. Since the autocorrelation is adequate for classifying the data, higher-order statistics are not useful in this example. In other examples, perhaps, when the data are contaminated by lot of Gaussian noise, AR estimates based on third- or fourth-order cumulants may be more useful.

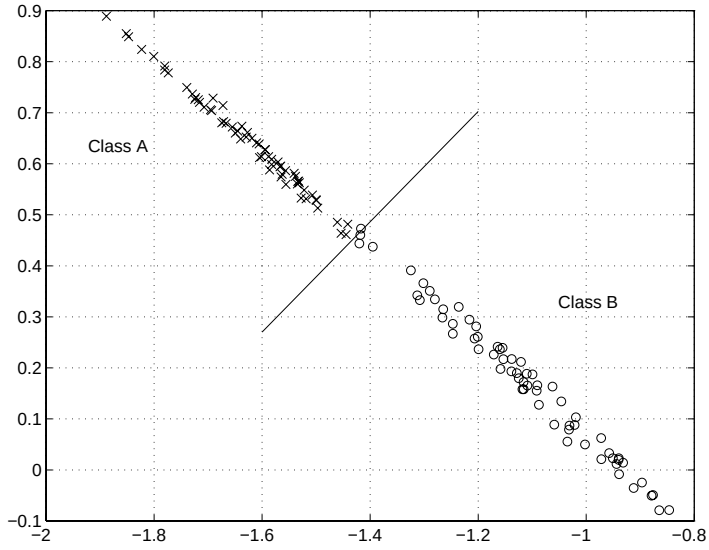


Figure 1-42 Acoustic Data: AR(2) Coefficients

Laughter Data

Speech signals are highly nonstationary, and are known to exhibit phase- and frequency-coupling phenomena. Here we will use some of the Higher-Order Spectral Analysis Toolbox M-files to illustrate these features. We will use the initial 1400 sample segment of the data in the file `laughter.mat`, which is included in the standard MATLAB distribution package. The commands that we used to generate the various figures, and to estimate various quantities are listed in Table 1-4.

Figure 1-43 shows the data and the histogram; the univariate distribution does not appear to be symmetrical. The mean, standard deviation, skewness, and kurtosis were estimated to be 0.5621, 536.69, 0.1681, and 1.3277 indicating that the data are non-Gaussian, and that the univariate pdf is not symmetric distributed.

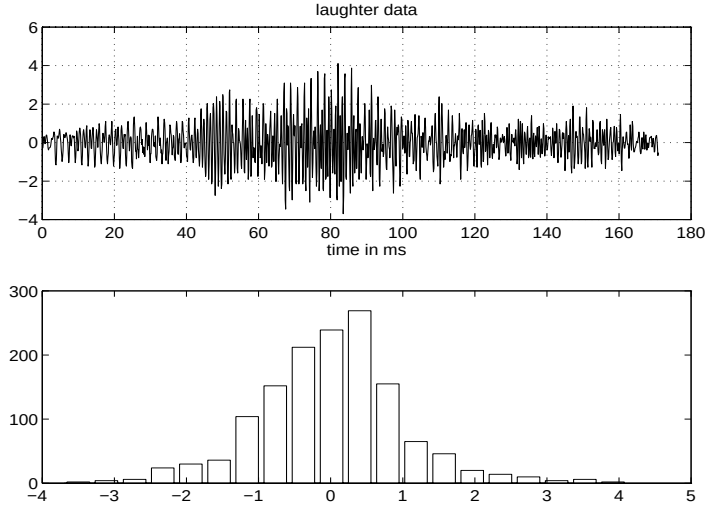


Figure 1-43 Laughter Data

Figure 1-44 shows the spectrogram or the STFT computed via `specgram`; we used an FFT length of 512, Hanning window of length 256, and an overlap of 240 samples. Three dominant frequency tracks can be seen in the figure, approximately around 550 Hz, 1100 Hz and 1550 Hz; the last *formant* begins around 30 ms; additional fragments are visible around 1800 Hz and 2100 Hz.

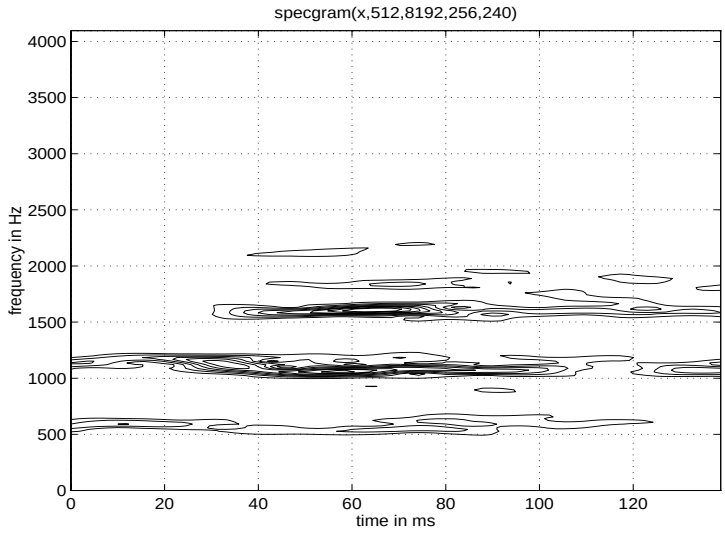


Figure 1-44 Laughter Data: Spectrogram

The spectrogram indicates that the speech signal is essentially harmonic in nature; it also shows that the frequencies appear to be approximately harmonically related. Since the STFT is resolution limited, we can use parametric methods such as `harmest` to obtain accurate estimates; we should, of course, keep in mind the nonstationary nature of the data.

The (1,1) panel of Figure 1-45 shows the singular values of the covariance matrix, as estimated by `harmest`; we set `nlag`=25 and `nfft`=512. The singular values are essentially zero after p = 12; hence we selected an order of 12, and obtained the various estimates shown in the figure.

We can use `roots` to compute the roots of the AR polynomials estimated by the AR and Min-Norm methods; `angle(roots(ar))/(2*pi)` yields the normalized frequencies and `abs(roots(ar))` yields the radius; the top two panels of Figure 1-47 show the locations of the roots of the two AR polynomials. The roots closest to the unit circle were found to be approximately at 0.0702, 0.1329, 0.1962 Hz for the AR method (less dominant roots at 0.2480, 0.3706 and 0.2012); for the Min-Norm method, the dominant roots are at frequencies 0.0710, 0.1945, 0.1272 and 0.1377. (These frequencies are normalized, that is, to find the correct frequencies, we should multiply them by the sampling frequency, which was 8000 Hz.)

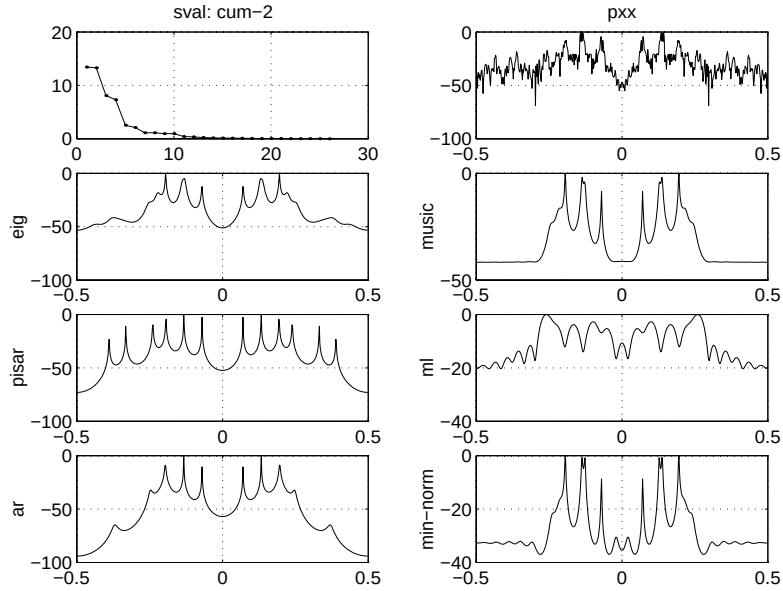


Figure 1-45 Laughter Data: "Power Spectra"

We can also use fourth-order cumulants to estimate the frequencies. Notice that the singular values of the cumulant matrix, shown in the (1,1) panel of Figure 1-46, look quite different from the singular values of the covariance matrix shown in the (1,1) panel of Figure 1-45; in particular, the singular values of the cumulant matrix appear to be essentially zero after $p = 8$; a possible explanation is that some of the harmonics are well-modeled as narrow-band Gaussian, and some as narrow-band non-Gaussian. Various "spectra" corresponding to order $p = 8$ are shown in Figure 1-46; the root locations of the AR polynomials are shown in the bottom panels of Figure 1-47. The dominant roots of the AR polynomial in the AR method are located at 0.1297 and 0.1964 Hz; a less dominant pole is at 0.2678 Hz. In the case of the Min-Norm method, the dominant roots are located at 0.0616, 0.1307, 0.1960 and 0.2622, suggesting that the signal has harmonics at approximately $2f_o$, $3f_o$, and $4f_o$.

We can test whether the bispectrum of the data are statistically nonzero by using `glstat`. The test results, using `cparm`=0.51, `nfft`=256 were: Test statistic for Gaussianity is 71.3231 with df = 48, Pfa = 0
Linearity test: R (estimated) = 2.3216, lambda = 2.376, R (theory) = 4.472, N = 14.

Since the Pfa is close to 0, we can be virtually certain that the data have nonzero bispectrum and hence are non-Gaussian. The estimated and theoretical R values are not close to each other indicating that the data are not linear.

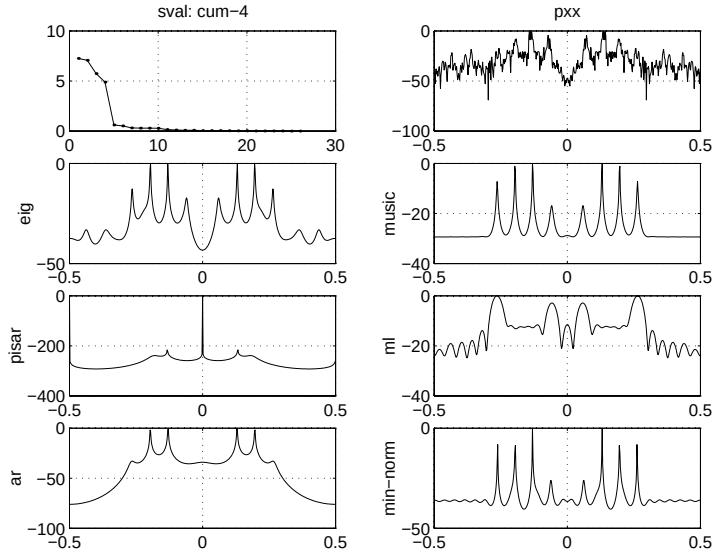


Figure 1-46 Laughter Data: “Cumulant Spectra”

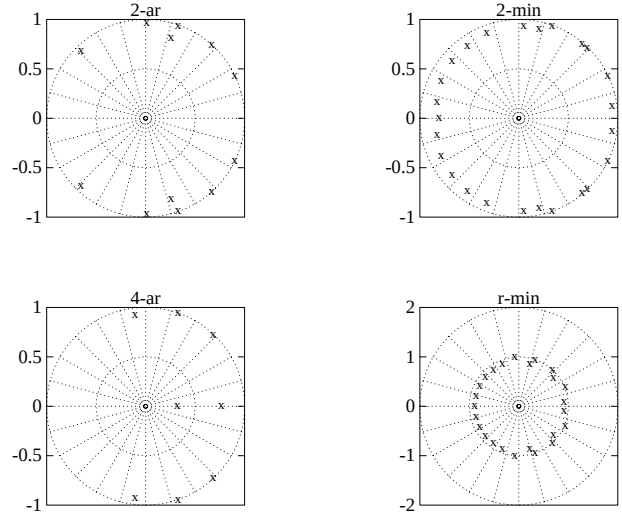


Figure 1-47 Laughter Data: “Root Locations”

In order to check for quadratic frequency coupling, we can estimate the nonparametric bispectrum (either via `bispeci` or `bispeci`) or the parametric bispectrum (via `qpctor`). The bispectrum estimated via `bispeci`, with `nfft`=256, `segsamp`=100, `overlap`=0, `wind`=1 is shown in Figure 1-48. Since the data contain several harmonic components, the bispectrum displays several impulses; in the case of speech signals, we know that these harmonics are at integer multiples of a fundamental. In this case, it suffices to examine the diagonal slice of the bispectrum that is shown in Figure 1-49. We used `pickpeak` to locate the dominant peaks, which were found to be at 0.0664, 0.1289 and 0.1953 Hz; given the FFT length of 256, we can conclude that the these three harmonics are quadratically frequency coupled; the frequency coupling is also evident (but less obvious) from the power spectral estimates and the spectrogram.

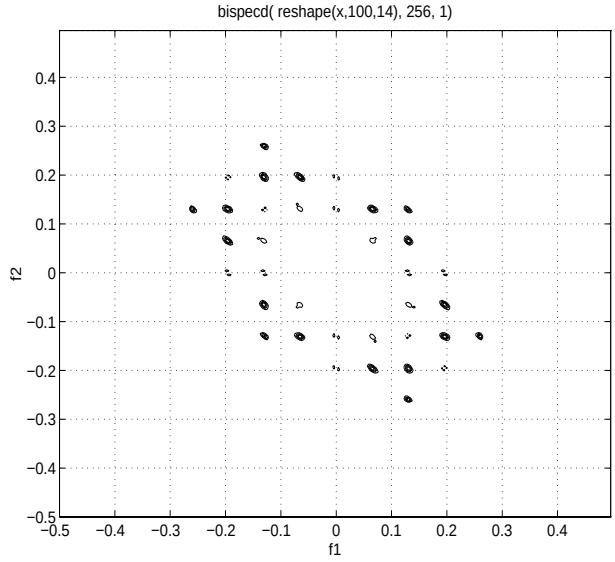


Figure 1-48 bispectrum – nonparametric

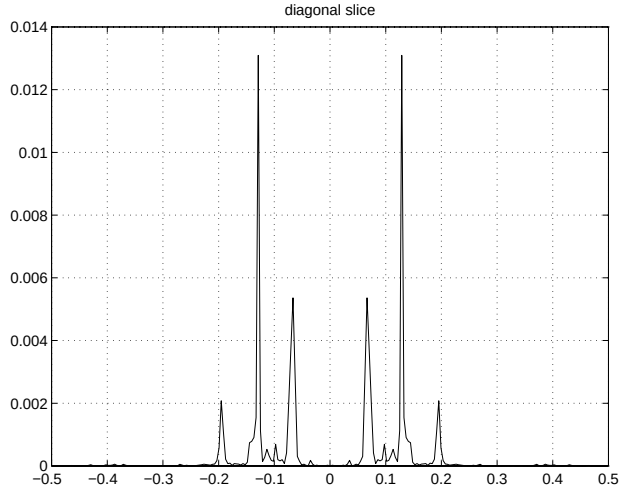


Figure 1-49 Diagonal Slice of bispectrum

We used routine `qpctor` to confirm the quadratic frequency coupling; we let nlags=25, segsamp=100, overlap=0; the singular values, shown in Figure 1-50, appear to fall in four clusters; we used $p = 10$ to estimate the parametric bispectrum shown in the bottom panel. The dominant peak is at (0.0625,0.0625) Hz, indicating the presence of a second harmonic at 0.1250 Hz; also note that the contour plot indicates peaks at (f_o, kf_o), k = 1,2,3, where $f_o \approx$ 0.0625 Hz. In the case of *self-coupling* [73] the bispectrum can be used to verify the presence of higher-order couplings as well (we have $f_o + f_o = 2f_o$, and $f_o + 2f_o = 3f_o$, $f_o + 3f_o = 4f_o$, etc., which lead to the indicated peaks).

In this example, we used `glstat` to confirm that the data are non-Gaussian, and nonlinear. We used `harmest` and `qpctor` to estimate the frequencies of the fundamental and the harmonics and to confirm frequency coupling. We used `roots` to determine the frequencies from the parameter vector estimated via the AR method; we can use `pickpeak` to locate the peaks in the various spectra estimated by `harmest`; note that the accuracy of the peak locations is limited by the FFT length (this should not be confused with resolving power). Since the signal is rich in harmonics, we found it more useful to look at a slice (the diagonal slice) of the bispectrum. Our final conclusions are that the speech segment in question is non-Gaussian, nonlinear, and contains harmonics of the form $f_o k = kf_o$, with $f_o \approx 500$ Hz.

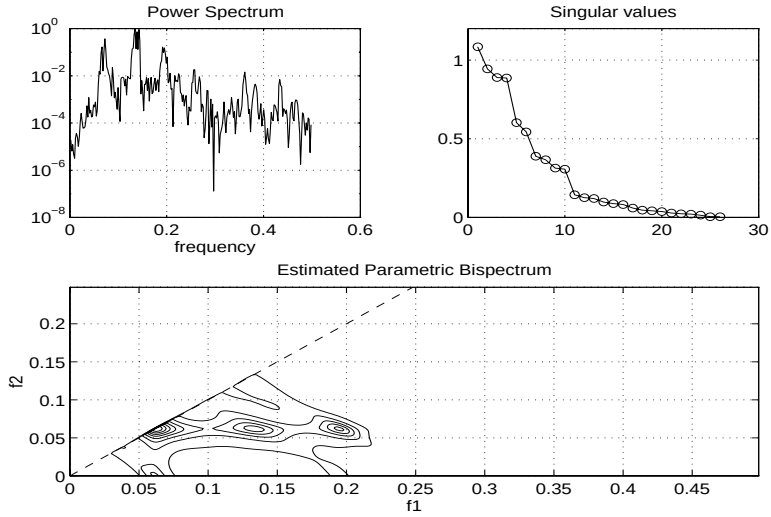


Figure 1-50 Parametric bispectrum via qpctor

Table 1-4: Commands Used to Analyze Laughter Data

```
load laughter mat                % laughter data
y = y(1:1400);
sp = (y-mean(y))/std(y);         % standardize

%
figure(1)           % data and histograms:
subplot(211), plot(1:length(sp),sp),grid,
subplot(212), hist(sp), grid

%    % summary stats:
c1 = mean(sp);                   c2 = cumest(sp,2);
c3 = cumest(sp,3) / c2^(3/2);    c4 = cumest(sp,4) / c2^2;
fprintf(' Mean                     %g\n',c1);
fprintf(' Variance                 %g\n',c2);
fprintf(' Skewness (normalized)    %g\n', c3);
fprintf(' Kurtosis (normalized)    %g\n', c4);

figure(2),  specgram(sp,512,8000,hamming(256),240);

%      -------- power spectra and cum-4 spectra
figure(3),  [px2,a21,a22]=harmest(sp,25,12,'biased',512,2);
figure(4),  [px4,a41,a42]=harmest(sp,25,8,'biased',512,4);

r21 = roots(a21); r22 = roots(a22);
r41 = roots(a41); r42 = roots(a42);
figure(5)
subplot(221), polar(angle(r21), abs(r21), 'x'), grid
subplot(222), polar(angle(r22), abs(r22), 'x'), grid
subplot(223), polar(angle(r41), abs(r41), 'x'), grid
subplot(224), polar(angle(r42), abs(r42), 'x'), grid

glstat(sp,0.51.256);             % gl test

%      -------- bispectrum and diagonal slice

figure(6),        [Blaf,w] =  bispecd(sp,256,1,100,0);
figure(7),        dB = abs(diag(Blaf));
plot(w,dB),       grid, title('diagonal slice')
[loc,val] = pickpeak(dB,3)
disp(w(loc))

figure(8),  qpctor(sp,25,10,512,100,0);   % QPC test
```

Pitfalls and Tricks of the Trade

As the previous examples illustrate, we should first use univariate statistics (mean, median, skewness, kurtosis, histograms) and second-order statistics (SOS). Before using algorithms based on higher-order statistics, we should test the data for Gaussianity, and if applicable, for linearity as well. Note that the symmetry of the univariate pdf does not imply that the bispectrum is zero.

Model parameters (e.g., ARMA, harmonics) estimated by algorithms based on cumulants of different orders need not agree with one another. For example, in the case of the *laughter* data, we saw that the second- and fourth-order *spectra* were different; a possible reason could be that some of the harmonic components can be well-modeled as narrow-band Gaussian.

Similarly, in the case of ARMA modeling, algorithms based on cumulants of different orders may lead to different results. Consistent results may be expected only in the case of noise-free data and long data lengths. Consider the following three-component model

$$y(n) = \sum_{i=1}^{3} x_i(n)$$
$$x_i(n) = \sum_{i=1} h_i(k) u_i(n-k).$$

Assume that $u_1(n)$ is i.i.d. Gaussian, $u_2(n)$ is i.i.d., single-sided exponential, and $u_3(n)$ is i.i.d., and double-sided exponential (Laplace); further assume that the u_i's are independent of one another. Let σ_i^2 denote the variance, and $\gamma_{3,i}$ and $\gamma_{4,i}$ the skewness and kurtosis of $ui(n)$. Then,

$$S_{2x}(\omega) = \sigma_1^2 \left|H_1(\omega)\right|^2 + \sigma_2^2 \left|H_2(\omega)\right|^2 + \sigma_3^2 \left|H_3(\omega)\right|^2$$
$$S_{3x}(\omega_1, \omega_2) = \gamma_{3,2} H_2(\omega_1) H_2(\omega_2) H_2^*(\omega_1 + \omega_2)$$
$$S_{4x}(\omega_1, \omega_2, \omega_3) = \gamma_{4,2} H_2(\omega_1) H_2(\omega_2) H_2(\omega_3) H_2^*(\omega_1 + \omega_2 + \omega_3)$$
$$+ \gamma_{4,3} H_3(\omega_1) H_3(\omega_2) H_3(\omega_3) H_3^*(\omega_1 + \omega_2 + \omega_3)$$

In this example, the power spectrum is the sum of the power spectra of the three components; since $u_1(n)$ and $u_3(n)$ are both symmetric distributed, their skewnesses are zero; hence, only the second component contributes to the bispectrum; finally, since $u_1(n)$ is Gaussian, all of its cumulants of order greater than two are zero; hence, it does not contribute to the fourth-order cumulant.

Even though $\gamma_{3,2} \neq 0$ in the above example, it is easy to construct examples of $H_2(\omega)$ so that the bispectrum is zero [63]. Hence, a zero bispectrum is not inconsistent with nonzero skewness of the driving i.i.d. process.

In the context of harmonic processes of the form,

$$s(n) = \sum_{k=1}^{p} \alpha_k \cos(\omega_k n + \phi_k),$$

the phrases *frequency coupling* and *phase coupling* are often used in the same sense; the former refers to relationships of the form $\omega_3 = \omega_2 + \omega_1$, and the latter to $\phi_3 = \phi_2 + \phi_1$. Generally, but not always, the two relationships go hand in hand.

Further, in the case of harmonic processes, the power spectrum and parametric methods based on the autocorrelation are usually sufficient, unless the data contain narrow-band Gaussian components. Higher-order cumulants and spectra are useful to isolate specific types of coupling (quadratic, cubic, etc.).

Several of the M-files in the Higher-Order Spectral Analysis Toolbox allow the user to segment the data, that is, cumulants are estimated for each segment and are then averaged across the set of segments; such segmentation usually speeds up calculations, at a slight loss in statistical efficiency; note that segmented estimates can never be better than unsegmented estimates.

In the case of bispectra, as we have seen earlier, we can obtain consistent estimates by either estimating bispectra from segments and averaging them (frequency resolution limited by the segment length) or by applying an appropriate smoothing window (in the frequency, data or lag domain).

Be aware that the higher-order moments and cumulants of complex processes can be defined in different ways, and that their polyspectra do not possess all the symmetry properties of their real counterparts.

There has been lot of confusion about the nonredundant region (NRR) of the bispectrum; we refer the reader to Brillinger [4] for an explanation of why the NRR is the triangle with vertices (0,0), (0,π), and (2π/3,2π/3), and not the triangle with vertices (0,0), (0,π), and (π/2,π/2).

Finally, it is easy to be misled by contour plots, where `100*eps` may stand out as a peak in a background of `eps`; careful attention should be paid to scaling the data; normalizing the data to unity variance is usually helpful.

Data Files

The Higher-Order Spectral Analysis Toolbox distribution diskette includes several `.mat` files. These files are used in the "Tutorial" to demonstrate various toolbox functions. Descriptions of these `.mat` files are given below.

`ar1.mat`	AR(2) synthetic
`arma1.mat`	ARMA(2,1) synthetic
`doa1.mat`	Synthetic for the direction of arrival problem
`gldat.mat`	Data for Gaussianity/linearity tests
`harm.mat`	Synthetic for the harmonics in noise problem
`ma1.mat`	MA(3) synthetic
`nl1.mat`	Synthetic for testing the nonlinear routines
`nl2.mat`	Synthetic for testing the nonlinear routines
`qpc.mat`	Synthetic for the quadratic phase coupling problem
`riv.mat`	Synthetic for the adaptive LP problem
`tde1.mat`	Synthetic for the time-delay estimation problem
`tprony.mat`	Synthetic for testing Prony's method
`wigdat.mat`	Synthetic for testing the Wigner routines

In the following, signal-to-noise ratio (SNR) is defined as the ratio of the signal variance to the noise variance; when expressed in dB, it is given by $10\log_{10}(\sigma_s^2/\sigma_n^2)$, where σ_s^2 is the variance of the signal, and σ_n^2 is the variance of the noise. All frequencies are in Hz relative to a nominal sampling frequency of 1 Hz.

`ar1.mat` An AR (2) synthetic; the AR parameters are [1, –1.5, 0.8]; the time-series length is 1024 samples; input distribution is single-sided exponential; additive white Gaussian noise was added so as to obtain a SNR of 20 dB. The vector y contains the AR(2) time-series.

`arma1.mat` An ARMA (2,1) synthetic; the AR parameters are [1, –0.8, 0.65]; the MA parameters are [1, –2]; the time-series length is 1024 samples; input distribution is single-sided exponential; additive white Gaussian noise was added so as to obtain a SNR of 20 dB. The vector `y` contains the ARMA(2,1) synthetic. The model is mixed-phase.

`doa1.mat` This data file contains the 4096 -by- 8 matrix `ymat`. It is a synthetic for the DOA problem, with two sources at bearings, $\theta_1 = -15°$ and $\theta_2 = -25°$; the eight sensors are spaced half a wavelength apart. The source signals are Laplace distributed, and have unity variance. Spatially correlated Gaussian noise, with unity variance, was added to obtain the noisy signals. The spatial color is such that the angular noise spectrum of the noise shows sharp peaks at ±30°.

`gldat.mat` This file contains the vectors `e, g, l, u, x,` and `z`; each vector consists of 512 samples. Sequences `e, g, l,` and `u` are realizations of i.i.d. sequences with exponential, Gaussian, Laplace, and uniform distributions. Sequence `x` is obtained by passing **e** through the AR filter, [1, –1.5, 0.8]; and, $z(n) = x^3(n)$.

`harm.mat` A harmonics-in-noise synthetic. Two unity amplitude harmonics, with frequencies $\lambda_1 = 0.1$ and $\lambda_2 = 0.2$, were corrupted with additive colored Gaussian noise with a variance of 0.5. Since each of the harmonics has power 0.5, the SNR is 3 dB. Thirty-two independent realizations, each with 128 samples were generated. The phases of the harmonics were chosen randomly for each realization. The colored Gaussian noise was generated by passing white Gaussian noise through an AR filter with coefficients [1, –1.058, 0.81]. The noise spectrum has a pole at 0.15 Hz with a damping factor of 0.9. The set of realizations is contained in the matrix `zmat`; the columns correspond to independent realizations.

`ma1.mat` An MA(3) synthetic; the MA parameters are [1,0.9,0.385, –0.771]; the time-series length is 1024 samples; input distribution is single-sided exponential; additive white Gaussian noise was added so as to obtain a SNR of 20 dB. The vector `y` contains the MA(3) synthetic. The model is mixed-phase.

`nl1.mat` This file contains data to test functions `nlpow` and `nltick`. A white Gaussian process, x, is passed through a second-order Volterra system, to obtain the output process y. Matrices `x` and `y` are 64×64, with columns corresponding to realizations. The input-output relationship for a second-order Volterra system is given by,

The filter h was chosen to be the IR of an FIR(12) filter, with a cutoff of 0.2Hz, and was generated with h = fir1(11,0.4). The filter q was chosen such that

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) + \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} q(k,l)x(n-k)x(n-l).$$

$q(k,l) = h1(k)h1(l)$, where $h1$ is the IR of an FIR(12) filter, with a cutoff of 0.1Hz, and was generated by `h1 = fir1(11,0.2)`. The output data were generated using function `nlgen`, `y = nlgen (x,h,q)`. The impulse responses (IR) and transfer functions (TF) of the linear part (`h`), and the quadratic part (`q`) are shown in Figure 1-51.

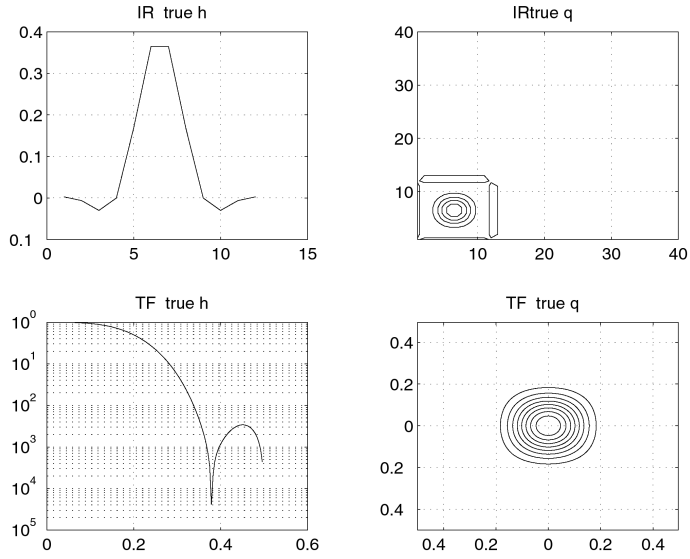


Figure 1-51 Transfer Functions of a Second-Order Volterra Model

`nl2.mat` This file contains data to test function `nlpow`. This data file was generated in the same way as was `nl1.mat`, except that the process x was chosen to be i.i.d. and Laplacian.

`qpc.mat` A quadratically phase coupled synthetic. The data consist of four harmonics corrupted by white Gaussian noise. The frequencies of the harmonics are $\lambda_1 = 0.1$, $\lambda_2 = 0.15$, $\lambda_3 = 0.25$ and $\lambda_4 = 0.40$ Hz. For each realization, the phases of the first, second, and fourth harmonics, ϕ_1, ϕ_2, and ϕ_4 were chosen randomly; the phase of the third harmonic was set to $\phi_3 = \phi_1 + \phi_2$. Note that the triplet of harmonics with frequencies (0.1, 0.15, 0.25) exhibits both phase and frequency coupling; the triplet with frequencies (0.15, 0.25, 0.40) is frequency-coupled, but not phase-coupled. Sixty-four independent realizations, each consisting of 64 samples, were generated. The amplitudes of all four harmonics were unity, and white Gaussian noise with a variance of 1.5 was added to the signal. The set of realizations is contained in the matrix `zmat`; the columns correspond to independent realizations.

`riv.mat` This data file contains three vectors, `y`, `zw`, and `zc`, each consisting of 1024 samples. A zero-mean exponentially distributed i.i.d. sequence was passed through the AR filter [1, –1.5, 0.8] to obtain `y`. Additive white Gaussian noise was added to **y** to obtain the noisy signal `zw` whose SNR is 10 dB. Colored Gaussian noise, generated by passing a white Gaussian sequence through the AR filter [1, 0, 0.49] was added to **y** to obtain `zc`, also at a SNR of 10 dB.

`tde1.mat` A synthetic for time-delay estimation. The signal at the first sensor is i.i.d., zero mean, single-sided exponentially distributed with unity variance, and skewness of two. The signal at the second sensor is a delayed version of the signal at the first sensor (delay = 16 samples). The first signal was corrupted by white Gaussian noise to obtain a SNR of 0 dB. The signal at the second sensor was corrupted by colored Gaussian noise, obtained by passing the noise at the first sensor through the MA filter, [1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1]. Note that the two noise signals have a strong spatial correlation at a delay of five samples. The signal length is samples. Vectors `s1` and `s2` contain the simulated signals at the two sensors.

`tprony.mat` This mat-file contains the 256 sample complex transient `x`, and is used to test the function `hprony`; here,

$$x(n) = \exp\{j\pi/2 + n(-0.1 + j0.84\pi)\} + 2\exp\{j\pi/4 + n(-0.2 + j0.64\pi)\}$$

`wigdat.mat` This file contains four signals, `s1`, `s2`, `s3`, and `s4`, described below, which are used to test the functions `wig2`, `wig2c`, `wig3`, `wig3c`, `wig4`, and `wig4c`.

Signal `s1` is a real harmonic, given by

$$s1(n) = \cos(2\pi f n), \quad f = 0.1$$

Signals s2 and s3 are generated via

$$s(n) = \exp\left(-2\pi\left(\frac{T(n-n_0)}{2\sigma}\right)^2\right)\cos(2\pi f n T)$$

with $T = .001$, $n_0 = 20$, $f = 50$, $\sigma = 0.01$ for signal s2, and $T = .001$, $n_0 = 50$, $f = 150$, $\sigma = 0.01$ for signal s3. Signal s4 is the sum of s2 and s3. Note that signals s2 and s3 are harmonics at frequencies 0.05 and 0.15 Hz, which have been multiplied by a Gaussian shaped window function. Signals **s1**, s2, s3, and s4 are displayed in Figure 1-52.

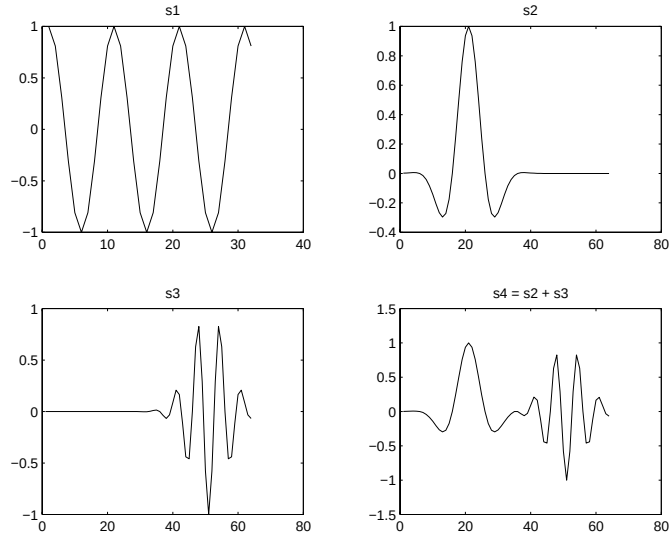


Figure 1-52 Signals in file wigdat.mat

References

1 Allen, J.B., "Short-time spectral analysis, synthesis and modification by discrete Fourier transform," *IEEE Trans ASSP*, Vol. 25, pp. 235-38, June 1977.

2 Blackman, R.B. and J.W. Tukey, *The Measurement of Power Spectra*, New York: Dover, 1959.

3 Boudreaux-Bartels, G., *Time-Frequency Signal Processing Algorithms: Analysis and Synthesis Using Wigner Distributions*, Ph.D. dissertation, Rice University, 1984.

4 Brillinger, D.R., "An Introduction to Polyspectra," *Ann. Math. Stat.*, Vol. 36, pp. 1351-74, 1965.

5 Brillinger, D.R., *Time Series: Data Analysis and Theory*, McGraw Hill, 1981.

6 Brillinger, D.R. and M. Rosenblatt, "Computation and interpretation of k-th order spectra," in *Spectral Analysis of Time Series*, B. Harris, ed., pp. 189-232, John Wiley, 1967.

7 Cadzow, J.A., "Spectral Estimation: An Overdetermined Rational Model Equation Approach," *Proc. IEEE*, Vol. 70, pp. 907-38, 1982.

8 Capon, J., "High-resolution frequency-wavenumber analysis," *Proc. IEEE*, Vol. 57, pp. 1408-18, 1969.

9 Chandran, V. and S. Elgar. "A general procedure for the derivation of principal domains of higher order spectra." *IEEE Trans. on Signal Processing*, SP-42: pp. 229-33, Jan. 1994.

10 Choi, H. and W.J. Williams, "Improved Time-Frequency Representation of Multicomponent Signals Using Exponential Kernels," *IEEE Trans. ASSP*, pp. 862-71, June 1989.

11 Classen, T.A.C.M. and W.F.G. Mecklenbraüker, "The Wigner distribution — a tool for time-frequency signal analysis," *Phillips J. Res.*, Vol. 35, pp. 217-50, 276-300, 372-89, 1067-72, 1980.

12 Classen, T.A.C.M. and W.F.G. Mecklenbraüker, "The aliasing problem in discrete-time Wigner distributions," *IEEE Trans ASSP*, pp. 1067-72, Oct. 1983.

13 Cohen, L., "Time-Frequency Distributions: A Review," *Proc. IEEE*, pp. 941-81, July 1989.

14 Fonollosa, J.R. and C.L. Nikias, "Wigner Higher-Order Moment Spectra: Definitions, Properties, Computation and Applications to Transient Signal Detection," *IEEE Trans. SP*, Jan. 1993.

15 Fonollosa, J.R. and C.L. Nikias, "Analysis of finite-energy signals using higher-order moments- and spectra-based time-frequency distributions," *Signal Processing*, Vol. 36, pp. 315-28, 1994.

16 Friedlander, B. and B. Porat, "Asymptotically optimal estimation of MA and ARMA parameters of non-Gaussian processes from high-order moments," *IEEE Trans. on Automatic Control*, Vol. 35(1), pp. 27-37, 1990.

17 Gabor, D., "Theory of communication," *J.i.e. E.*, Vol. 93, pp. 429-59, 1946.

18 Gerr, N.L., "Introducing a third-order Wigner distribution," *Proc. IEEE*, pp. 290-92, Mar. 1988.

19 Giannakis, G.B. and J.M. Mendel, "Identification of non-minimum phase systems using higher-order statistics," *IEEE Trans. ASSP*, Vol. 37, pp. 360-77, Mar. 1989.

20 Giannakis, G.B. and J.M. Mendel, "Cumulant-based order determination of non-Gaussian ARMA models," *IEEE Trans. ASSP*, pp. 1411-21, Aug. 1990.

21 Golub, G.H. and C.F. Van Loan, *Matrix Computations*, pp. 420-5, Baltimore: The Johns Hopkins University Press, 1983.

22 Hasan, T., "Nonlinear time series regression for a class of amplitude modulated sinusoids," *J. Time Series Analysis*, Vol. 3, pp. 2, 109-22, 1982.

23 Haykin, S., *Adaptive Filter Theory*, New Jersey: Prentice-Hall, pp. 391-94, 198-205, 464-70, 234-36, 2nd ed., 1991.

24 Hinich, M.J., "Testing for Gaussianity and linearity of a stationary time series," *J. Time Series Analysis*, Vol. 3, pp. 169-76, 1982.

25 Hlaswatch, F., and G.F. Boudreaux-Bartels, "Linear and Quadratic Time-Frequency Representations," *IEEE Signal Processing Magazine*, pp. 21-67, Apr. 1992.

26 Johnson, D.H., "The application of spectrum estimation methods to bearing estimation problems," *Proc. IEEE*, Vol. 70, pp. 975-89, 1982.

27 Kay, S.M., *Modern Spectral Estimation: Theory and Applications*, New Jersey: Prentice-Hall, 1988.

28 Kay, S.M. and S.L. Marple, "Spectrum Analysis — A Modern Perspective," *Proc. IEEE*, Vol. 69, pp. 1380-1418, 1981.

29 Krauss, T., J.N. Little, and L. Shure, *MATLAB Signal Processing Toolbox User's Guide*, The MathWorks Inc., 1994.

30 Kumaresan, R. and D.W. Tufts, "Estimating the angles of arrival of multiple plane waves," *IEEE Trans. AES*, Vol. 19, pp. 134-39, 1983.

31 Kung, S.Y. et al, "State-space and SVD approximation methods for the harmonic retrieval problem," *J. Opt. Soc. Amer.*, Vol. 73., pp. 1799-1811, 1983.

32 Makhoul, J., "Stable and efficient lattice methods for linear prediction," *IEEE Trans. ASSP*, Vol. 25, pp. 423-28, Oct. 1977.

33 Mallat, S., "A theory for multi-resolution signal representation: the wavelet transform," *IEEE Trans. PAMI*, Vol. 11, pp. 674-93, July 1989.

34 Marple, S.L., Jr., *Digital Spectral Analysis with Applications*, pp. 229-31, New Jersey: Prentice-Hall, 1987.

35 Marple, Jr., "A new autoregressive spectrum analysis algorithm," *IEEE Trans. ASSP*, Vol. 28, pp. 441-50, Aug. 1990.

36 Matsuoka, T. and T.J. Ulrych, "Phase estimation using the bispectrum," *Proc. IEEE*, Vol. 72, pp. 1403-11, 1984.

37 Mendel, J.M., "Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications," *Proc. IEEE*, Vol. 79, pp. 278-305, 1991.

38 Nikias, C.L. and J.M. Mendel, "Signal processing with higher-order spectra," *IEEE Signal Processing Magazine*, Vol. 10, No 3, pp. 10-37, July 1993.

39 Nikias, C.L. and R. Pan, "Time delay estimation in unknown Gaussian spatially correlated noise," *IEEE Trans. ASSP*, Vol. 36, pp. 1706-14, Nov. 1988.

40 Nikias, C.L. and A. Petropulu, *Higher-Order Spectra Analysis: A Nonlinear Signal Processing Framework*, New Jersey: Prentice-Hall, 1993.

41 Nikias, C.L. and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

42 Oppenheim, A.V. and R.W. Schafer, *Digital Signal Processing*, New Jersey: Prentice-Hall, 1989.

43 Pan, R. and C.L. Nikias, "The complex cepstrum of higher-order cumulants and non-minimum phase system identification," *IEEE Trans ASSP*, Vol. 36, pp. 186-205, Feb. 1988.

44 Pan, R. and C.L. Nikias, "Harmonic decomposition methods in cumulant domains," *Proc. ICASSP-88*, pp. 2356-59, New York, 1988.

45 Patel, J.K. and C.B. Read, *Handbook of the Normal Distribution*, Sec 7.9.4, M. Dekker, New York, 1982.

46 Perryin, F. and R. Frost, "A unified definition for the discrete-time, discrete-frequency, and discrete time/frequency Wigner distributions," *IEEE Trans. ASSP*, Vol. 34, pp. 858-67, Aug. 1986.

47 Pflug, L.A., G. E. Ioup, J. W. Ioup and R. L. Field, "Properties of higher-order correlations and spectra for bandlimited transients," *J. Acoust. Soc. Am.*, Vol. 91(2), pp. 975-88, Feb. 1992.

48 Pisarenko, V.F., "The retrieval of harmonics from a covariance function," *Geophysical J. Royal Astron. Soc.*, Vol. 33, pp. 347-66, 1973.

49 Porat, B., *Digital Processing of Random Signals*, New Jersey: Prentice-Hall, 1994.

50 Porat, B., B. Friedlander, and M. Morf, "Square-root covariance ladder algorithms," in *Proc. ICASSP-81*, Atlanta, GA, pp. 877-880, March 1981.

51 Powers, E.J., Ch.P. Ritz, C.K. An, S.B. Kim, R.W. Miksad and S.W. Nam, "Applications of digital polyspectral analysis to nonlinear systems modeling and nonlinear wave phenomena," *Proc. Workshop on Higher-Order Spectral Analysis*, pp. 73-77, Vail, Colorado, June 1989.

52 Priestley, M.B., *Spectral Analysis and Time Series*, Academic Press, London, 1981.

53 Priestley,M.B., *Non-linear and non-stationary time series analysis*, Academic Press, London, 1988.

54 Raghuveer, M.R. and C.L. Nikias, "Bispectrum Estimation: A Parametric Approach," *IEEE Trans. ASSP*, Vol. 33, pp. 1213-30, 1985.

55 Rangoussi, M. and G.B. Giannakis, "FIR modeling using log-bispectra: Weighted least-squares algorithms and performance analysis," *IEEE Trans. Cir Sys*, Vol. 38, pp. 281-96, 1991.

56 Rioul, O. and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Magazine*, pp. 14-38, Oct 1991.

57 Roy, R. and T. Kailath, "ESPRIT – Estimation of signal parameters via rotational invariance techniques," *IEEE Trans ASSP*, Vol. 37, pp. 984-95, July 1989.

58 Stoica, P., T. Soderstrom and B. Friedlander, "Optimal instrumental variable estimates of the AR parameters of an ARMA process," *IEEE. Trans Auto. Control*, Vol. 30, pp. 1066-74, Nov 1985.

59 Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

60 Swami, A., *System Identification Using Cumulants*, Ph.D. Dissertation, University of Southern California, pp. 107-8, 1988.

61 Swami, A., "Higher-Order Wigner Distributions," in *Proc. SPIE-92, Session on Higher-Order and Time-Varying Spectral Analysis*, Vol. $1770, 290-301, San Diego, CA, July 19-24, 1992.

62 Swami, A., "Third-order Wigner distributions," *Proc. ICASSP-91*, pp. 3081-84, Toronto, Canada, May 1991.

63 Swami, A., "Some new results in higher-order statistics," *Proc. Intl. Signal Processing Workshop on Higher-Order Statistics*, Chamrousse, France, pp. 135-38, July 1991.

64 Swami, A., "Higher-order Wigner distributions," in *Proc. SPIE-92, Session on Higher-Order and Time-Varying Spectral Analysis*, Vol. 1770, pp. 290-301, San Diego, CA, July 1992.

65 Swami, A., "Pitfalls in polyspectra," *Proc. ICASSP-93*, Minneapolis, Vol. IV, pp. 97-100, Apr. 1993.

66 Swami, A., and J.M. Mendel, "Adaptive Cumulant-Based Estimation of ARMA Parameters," *Proc. Amer. Control Conf., ACC-88*, Atlanta, GA, pp. 2114-19, June 1988.

67 Swami, A. and J.M. Mendel, "Lattice algorithms for recursive instrumental variable methods," to appear in *The Int'l Journal of Adaptive Control and Signal Processing*, 1996. See also: *USC-SIPI Report-117*, University of Southern California, Los Angeles, Dec. 1987; *Proc. ICASSP-88*, New York, pp. 2248-51, Apr. 1988; *Proc. Amer. Control Conf.*, Atlanta, GA, pp. 2114-19, June 1988.

68 Swami, A. and J.M. Mendel, "ARMA parameter estimation using only output cumulants," *IEEE Trans. ASSP*, Vol. 38, pp. 1257-65, July 1990.

69 Swami, A. and J.M. Mendel, "Cumulant-based approach to the harmonic retrieval and related problems," *IEEE Trans. ASSP*, Vol. 39, pp. 1099-1109, May 1991.

70 Swami, A. and J.M. Mendel, "Identifiability of the AR parameters of an ARMA process using cumulants," *IEEE Trans. on Automatic Control*, Vol. 37, pp. 268-73, Feb. 1992.

71 Swami, A. and P. Tichavsky, "The n-th order moment (cumulant) of multiple sinusoids: strong ergodicity and related issues" *Proc. IEEE SP/ATHOS Workshop on HOS*, Girona, Spain, June 1995.

72 Tekalp and A. T. Erdem, "Higher-order spectrum factorization in one and two dimensions with applications in signal modeling and nonminimum phase system identification" *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 37(10), pp. 1537-49, 1989.

73 Tick, L.J., "The estimation of transfer functions of quadraticsystems," *Technometrics*, 3, 563-67, Nov 1961.

74 Tugnait, J.K., "Identification of linear stochastic systems via second- and fourth-order cumulant matching," *IEEE Trans. on Information Theory*, Vol. 33(3), pp. 393-407, May 1987.

75 Tugnait, J.K., "Approaches to FIR system identification with noisy data using higher-order statistics," *IEEE Trans. ASSP*, Vol. 38, pp. 1307-17, July 1990.

76 Zhou, G. and G.B. Giannakis, "Self coupled harmonics: stationary and cyclostationary approaches," *Proc. ICASSP*, pp. 153-156, Adelaide, Australia, Apr. 1994.

77 Zohar, S., "Toeplitz Matrix Inversion: the Algorithm of W.F. Trench," *J. Assoc. Comp. Mach.*, Vol. 16, pp. 592-601, 1968.

2

Reference

Function Tables

The following tables summarize the suite of M-files in the Higher-Order Spectral Analysis Toolbox by functionality. Reference entries for the functions are then listed in alphabetical order. On-line help is available via the `help` and `hosahelp` commands.

Higher-Order Spectrum Estimation: Conventional Methods

Function	Purpose
`cum2x`	Estimates second-order cross-cumulants
`cum3x`	Estimates third-order cross-cumulants
`cum4x`	Estimates fourth-order cross-cumulants
`cumest`	Estimates auto cumulants (orders 2,3,4)
`bicoher`	Estimates auto-bicoherence
`bicoherx`	Estimates cross-bicoherence
`bispecd`	Direct method for bispectrum estimation
`bispecdx`	Direct method for cross-bispectrum estimation
`bispeci`	Indirect method for bispectrum estimation
`glstat`	Detection statistics for Gaussianity and linearity tests

Higher-Order Spectrum Estimation: Parametric Methods

Function	Purpose
armaqs	ARMA parameter estimation, q-slice algorithm
armarts	ARMA parameter estimation, residual time series algorithm
armasyn	Generates ARMA synthetics
arorder	AR order estimation (AR and ARMA processes)
arrcest	AR parameter estimation
bispect	Computes theoretical bispectrum of an ARMA process
cumtrue	Computes true cumulants of ARMA processes
maest	MA parameter estimation
maorder	MA order estimation
rpiid	Generates sequence of i.i.d. random variables
trispect	Computes theoretical trispectrum of an ARMA process

Quadratic Phase Coupling (QPC)

Function	Purpose
qpcgen	Generates synthetics for the QPC problem
qpctor	Parametric QPC detection

Second-Order Volterra Systems

Function	Purpose
nlgen	Computes output of a second-order Volterra system
nlpow	Parameter estimation: arbitrary inputs: Powers' method
nltick	Parameter estimation: Gaussian inputs: Tick's method

Harmonic Retrieval

Function	Purpose
harmest	Estimates frequencies of sinusoidal signals
harmgen	Generates synthetics for harmonic retrieval problem

Time-Delay Estimation (TDE)

Function	Purpose
tde	TDE, based on cross-cumulants
tdeb	TDE, based on cross-bispectrum
tdegen	Generates synthetics for the TDE problem
tder	TDE, based on cross-correlation

Array Processing: Direction of Arrival (DOA)

Function	Purpose
doa	Estimates DOA from a linear array of sensors based on spatial cross-cumulant or covariance matrix
doagen	Generates synthetics for the DOA problem

Adaptive Linear Prediction

Function	Purpose
ivcal	Computes instrumental variable processes
rivdl	Adaptive LP using double lattice filter
rivtr	Adaptive LP using transversal filter

Impulse Response (IR), Magnitude and Phase Retrieval

Function	Purpose
biceps	Estimates IR complex cepstrum (lag domain)
bicepsf	Estimates IR & complex cepstrum (FFT method)
matul	Estimates Fourier phase and magnitude of a signal using the Matsuoka-Ulrych algorithm

Time-Frequency Estimates

Function	Purpose
wig2	Wigner spectrum
wig2c	Wigner spectrum, with Choi-Williams filtering
wig3	Wigner bispectrum
wig3c	Wigner bispectrum, with Choi-Williams filtering
wig4	Wigner trispectrum
wig4c	Wigner trispectrum, with Choi-Williams filtering

Utilities

Function	Purpose
`hosahelp`	One line help of Higher-Order Sectral Analysis Toolbox commands
`hprony`	Modeling of transient signals via Prony's method
`pickpeak`	Picks peaks subject to a separation criterion
`tls`	Total Least Squares solution
`trench`	Trench recursion for non-symmetric Toeplitz matrix

Demo

Function	Purpose
`hosademo`	Guided tour of the Higher-Order Sectral Analysis Toolbox

Miscellaneous

`help hosa` will display the list of HOSA Toolbox M-files, along with one-line descriptions. For additional help on a function, type `help function_name`. For example, to get help on function `doa`, type `help doa`.

Most of the arguments in the Higher-Order Sectral Analysis Toolbox M-files have default settings. Recall that arguments in a function have positional significance. Thus, if you want to specify the fourth argument of a function, you must specify (at least) the first four arguments. Function invocations of the form `function_name` (a, , , b) are invalid. Vectors must be specified within square brackets, e.g., [1,2,3]. Text or string variables must be specified within single quotes, that is, 'biased'. For more details, see the "Tutorial" section of the *MATLAB User's Guide*. For details on the algorithms and on the theory of cumulants and polyspectra, see the "Tutorial" section of this manual.

Prompting

Several Higher-Order Sectral Analysis Toolbox routines (such as `armasyn` and `qpcgen`) are interactive; the routines ask the user to specify various parameters; we refer to this as *prompting*.

Guided tour

Invoke the function `hosademo` for a guided tour of the Higher-Order Sectral Analysis Toolbox; demo will take you through all the examples in the manual.

Addenda

See the `Readme.m` file, or type `whatsnewhosa` at the MATLAB command prompt.

armaqs

Purpose

Estimates ARMA parameters using the q-slice algorithm

Syntax

```
[avec,bvec] = armaqs(y,p,q)
[avec,bvec] = armaqs(y,p,q,norder,maxlag,samp_seg,overlap,flag)
```

Description

`armaqs` estimates the ARMA parameters of the ARMA(p,q) process, `y`, using the q-slice algorithm. The AR parameters are estimated via the function `arrcest`; then the impulse response is estimated, and, finally, the MA parameters are estimated.

The AR order `p` must be greater than zero; function `maest` should be used in the $p = 0$ case; the MA order `q` must also be specified.

Variable `norder` specifies the cumulant-order to be used, and should be 3 or 4; the default value is 3.

`maxlag` specifies the maximum number of cumulant lags to be used; its default value is `p+q`.

Variables `samp_seg`, `overlap`, and `flag` control the manner in which sample cumulants are estimated:

`samp_seg` specifies the number of samples per segment; the default value is the length of the time series.

`overlap` specifies the percentage overlap between segments; the allowed range is [0,99]; and the default value is 0.

If `flag` is *biased*, then biased sample estimates are computed (default); if the first letter is not 'b', *unbiased* estimates are computed.

If `y` is a matrix, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension. Cumulants are estimated from each realization, and then averaged.

The estimated AR and MA parameters are returned in the vectors `avec` and `bvec`, respectively.

Algorithm

The signal is assumed to be described by

$$x(n) = -\sum_{k=1}^{p} a(k)x(n-k) + \sum_{k=0}^{q} b(k)u(n-k) = \sum_{k=0}^{\infty} h(k)u(n-k)$$

$$y(n) = x(n) + g(n)$$

where $u(n)$ is i.i.d. non-Gaussian, and $g(n)$ is Gaussian.

The AR parameters are obtained as the least-squares solution to the *normal* equations given by

$$\sum_{k=0}^{p} a(k)C_3(m-k, \rho) = 0, \quad m > q$$

$$\sum_{k=0}^{p} a(k)C_4(m-k, \rho, 0) = 0, \quad m > q$$

where $a(0) = 1$, $m = q+1, \ldots,$ `maxlag`, and $\rho = q-p, \ldots, q$. Since one must have at least $m = q+1, \ldots, q+p$ [1,2], it follows from the preceding normal equations, that the cumulant lags involved must include $q+1-p, \ldots, q+p$. The default value of `maxlag` follows from this.

The impulse response is then estimated via,

$$h(n) = \frac{\sum_{k=0}^{p} a(k)C_3(q-k, n)}{\sum_{k=0}^{p} a(k)C_3(q-k, 0)}, \quad n = 1, \ldots, q$$

or via,

$$h(n) = \frac{\sum_{k=0}^{p} a(k)C_4(q-k, n, 0)}{\sum_{k=0}^{p} a(k)C_4(q-k, 0, 0)}, \quad n = 1, \ldots, q$$

depending upon whether third- or fourth-order cumulants are used. We assume $b(0) = h(0) = 1$.

In our implementation, we estimate the AR and IR parameters simultaneously; and we use the Total Least Squares (TLS) solution (see `tls`).

The MA parameters are then obtained via,

$$b(n) = \sum_{k=0}^{p} a(k)h(n-k) = 0, \quad n = 1, \ldots, q.$$

See Also

`arrcest`, `maest`, `cumest`, `armarts`, `tls`

References

[1] Swami, A., and J.M. Mendel, "ARMA parameter estimation using only output cumulants," *IEEE Trans. ASSP*, Vol. 38, pp. 1257-65, July 1990.

[2] Swami, A. and J.M. Mendel, "Identifiability of the AR parameters of an ARMA process using cumulants," *IEEE Trans. on Automatic Control*, Vol. 37, pp. 268-73, Feb. 1992.

Purpose

Estimates ARMA parameters using the residual time series method

Syntax

```
[avec,bvec] = armarts(y,p,q)
[avec,bvec] = armarts(y,p,q,norder,maxlag,samp_seg,
   . . . overlap,flag)
```

Description

`armarts` estimates the ARMA parameters of the ARMA(p,q) process, `y`, via a three-step procedure. First, AR parameters are estimated (via the function `arrcest`); next, the MA(q) residual time series, $\tilde{y}(n) = \sum_{k=0}^{p} a(k)y(n-k)$, is created; and, finally, its MA parameters are estimated (via the function `maest`).

`p` and `q` are the AR and MA orders.

Variable `norder` specifies the cumulant-order to be used; `norder` should be ±3 or ±4; a negative value indicates that autocorrelations as well as cumulants of order `|norder|` should be used to estimate the AR parameters. The default value is 3.

`maxlag` specifies the maximum number of cumulant lags to be used; its default value is `q+p`.

Variables `samp_seg`, `overlap`, and `flag` control the manner in which sample cumulants are estimated:

`samp_seg` specifies the number of samples per segment; the default value is the length of the time series.

`overlap` specifies the percentage overlap between segments; the allowed range is [0,99]; and the default value is 0.

If `flag` is biased, then biased sample estimates are computed (default); if the first letter is not 'b', unbiased estimates are computed.

If `y` is a matrix, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension. Cumulants are estimated from each realization, and then averaged.

The estimated AR and MA parameters are returned in the vectors `avec` and `bvec`, respectively.

Algorithm

The signal is assumed to be described by

$$x(n) = -\sum_{k=1}^{p} a(k)x(n-k) + \sum_{k=0}^{q} b(k)u(n-k)$$

$$y(n) = x(n) + g(n)$$

where $u(n)$ is i.i.d. non-Gaussian, and $g(n)$ is Gaussian noise.

The AR parameters are obtained as the least-squares solution to the normal equations given by

$$\sum_{k=0}^{p} a(k)R(m-k) = 0, \quad m > q$$

$$\sum_{k=0}^{p} a(k)C_3(m-k, \rho) = 0, \quad m > q$$

$$\sum_{k=0}^{p} a(k)C_4(m-k, \rho, 0) = 0, \quad m > q$$

where $a(0) = 1$, $m = q + 1, \ldots,$ `maxlag`, and $\rho = q - p, \ldots, q$. Since one must have at least $m = q + 1, \ldots, q + p$ [1,2], it follows that the cumulant lags involved must include $q + 1 - p, \ldots, q + p$; this leads to the default value of `maxlag`.

If the estimated AR parameters are exact, then the residual time series is an MA(q) process,

$$\tilde{y}(n) \approx \sum_{k=0}^{p} a(k)y(n-k) = \sum_{k=0}^{q} b(k)u(n-k) + \sum_{k=0}^{p} a(k)g(n-k);$$

hence, the MA parameters can now be determined via the algorithms described in `maest`. Note that the even if $g(n)$ is white, the additive noise in $\tilde{y}(n)$ is no longer white. We assume that $b(0) = h(0) = 1$.

See Also

`arrcest`, `maest`, `cumest`, `armaqs`

References

[1] Giannakis, G.B. and J.M. Mendel, "Identification of non-minimum phase systems using higher-order statistics," *IEEE Trans. ASSP*, Vol. 37, pp. 360-77, Mar. 1989.

[2] Swami, A. and J.M. Mendel, "Identifiability of the AR parameters of an ARMA process using cumulants," *IEEE Trans. on Automatic Control*, Vol. 37, pp. 268-73, Feb. 1992.

armasyn

Purpose

Generates ARMA synthetics

Syntax

```
zmat = armasyn
zmat = armasyn(default)
```

Description

`armasyn` generates the time series described by

$$z(k) = \frac{B(z)}{A(z)}u(k) + g(k)$$

where u(k) is the i.i.d. input to the ARMA(p,q) model whose AR(p) and MA(q) polynomials are given by $A(z)$ and $B(z)$, respectively; $g(k)$ is signal-independent noise generated via

$$g(k) = \frac{B_n(z)}{A_n(z)}w(k)$$

where $w(k)$ is an i.i.d. sequence, and the polynomials $A_n(z)$ and $B_n(z)$ determine the spectrum of the noise.

If `armasyn` is invoked without any input arguments, then you are prompted for all the parameters (samples per realization; number of realizations; ARMA parameters for the signal process; pdf of the input driving noise, $u(k)$; noise variance; ARMA parameters for the observation noise process, $g(k)$; pdf of the noise, $w(n)$; signal-to-noise ratio).

If the function is invoked as `armasyn(default)`, where the variable `default` may take on *any* value(s), then, the default settings are used. Time series y in file `ar1.mat`, was generated via `y=armasyn(1);`.

`zmat` returns the generated data: each column corresponds to a different realization.

Purpose

Estimates the AR order of an AR or ARMA process

Syntax

```
p = arorder(y,norder,pmax,qmax,flag)
```

Description

Estimates the AR order of an AR or ARMA process using second-, third-, or fourth-order cumulants.

`y` is the observed ARMA process and *must* be a vector.

`norder` specifies the cumulant order(s) to be used; valid values are 2, ±3, and ±4; a value of –3 (–4) indicates that order determination should be based on the correlation as well as the third-order (fourth-order) cumulants. The default value is 3.

`pmax` specifies the maximum expected AR order; the default value is 10.

`qmax` specifies the maximum expected MA order; the default value is 10.

`flag` – if `flag` is 1, the internally chosen AR order is returned in `p`; otherwise, the plot of the singular values of the cumulant matrix is displayed on the graphics window, and you are prompted to choose the order. The default value is 1.

`p` is the estimated AR order.

Algorithm

Let $\bar{p}$ and $\bar{q}$ denote the maximum expected values of the AR and MA orders (the parameters `pmax` and `qmax`, respectively). For convenience, let $c_{ky}(m,\rho) :=$ cum$(y(n),y(n+m),y(n+\rho),y(n),\ldots,y(n))$, $k > 2$; note that we are suppressing (k – 3) of the lags, all of which are set to zero. Also let

$$\mathbf{c}_{ky}(m) := [\mathbf{c}_{ky}(m,-\bar{p})\ldots, \mathbf{c}_{ky}(m,\bar{q})]^T$$

Then, the singular values of the matrix,

$$\mathbf{C}_k = \begin{bmatrix} \mathbf{c}_{ky}(\bar{q}+1) & \ldots & \mathbf{c}_{ky}(\bar{q}+\bar{p}) \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{ky}(\bar{q}+\bar{p}+1) & \ldots & \mathbf{c}_{ky}(\bar{q}+2\bar{p}-1) \end{bmatrix}$$

are computed, where k = 2, 3, or 4. If `norder` is specified as –3(–4), the matrices $\mathbf{C}_2$ and $\mathbf{C}_3$ ($\mathbf{C}_4$) are concatenated.

Let $s(m)$ denote the singular values. The AR order p is then given by the value of n which maximizes $s(n) - s(n + 1)$, that is, it corresponds to the index at which the singular values show the maximum drop.

See Also

`maorder`

Reference

[1] Giannakis, G.B. and J.M. Mendel, "Cumulant-based order determination of non-Gaussian ARMA models," *IEEE Trans. ASSP*, pp. 1411-21, Aug. 1990

Purpose

Estimates AR parameters using the normal equations based on autocorrelations and/or cumulants

Syntax

```
avec = arrcest(y,p)
avec =arrcest(y,p,minlag,norder,maxlag,samp_seg,overlap,flag)
```

Description

`arrcest` estimates the AR parameters of the ARMA(p,q) process, `y`, via the normal equations based on autocorrelations and/or cumulants.

`p` is the AR order and must be specified.

`minlag` is the minimum value of m to be used in the normal equations. If you want to estimate the AR parameters of an ARMA(p,q) process, `minlag` should be greater than `q`. (See the Algorithm subsection for more details.)

The absolute value of `norder` specifies the cumulant order to be used; if `norder` is negative, least-squares solutions based on the simultaneous solution of both autocorrelation- and cumulant-based normal equations are obtained. The allowed values of `norder` are 2, ±3, ±4. The default value is 2.

`maxlag` specifies the maximum number of cumulant lags to be used; its default value is `p+minlag`.

Variables `samp_seg`, `overlap`, and `flag` control the manner in which sample cumulants are estimated:

`samp_seg` specifies the number of samples per segment; the default value is the length of the time series.

`overlap` specifies the percentage overlap between segments; the allowed range is [0,99]; and the default value is 0.

If `flag` is biased, then biased sample estimates are computed (default); if the first letter is not 'b', unbiased estimates are computed.

If `y` is a matrix, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension. Cumulants are estimated from each realization, and then averaged.

The estimated AR parameters are returned in the $p + 1$ element column vector `avec`.

Algorithm

In the noiseless case, the AR parameters are obtained as the least-squares solution to the normal equations given by

$$\sum_{k=0}^{p} a(k)R(m-k) = 0, \quad m > q$$

$$\sum_{k=0}^{p} a(k)C_3(m-k, \rho) = 0, \quad m > q$$

$$\sum_{k=0}^{p} a(k)C_4(m-k, \rho, 0) = 0, \quad m > q$$

The AR identifiability conditions, [1]-[2], require that $\rho = q_o - p, \ldots, q_o$, for any q_o, and $m = q + i + 1, \ldots, q + i + p$, $i \geq 0$; this leads to the default value of `maxlag`.

If the additive noise is white (Gaussian or non-Gaussian), the autocorrelation-based equations hold only for $m > \max(p,q)$. If the additive noise is non-Gaussian and white, the cumulant-based equations hold only for $m >$ `max(p,q)`; in these cases, choose `minlag > max(p,q)`.

See Also

`armarts`, `armaqs`, `cumest`

References

[1] Swami, A. and J.M. Mendel, "ARMA parameter estimation using only output cumulants," *IEEE Trans. ASSP*, Vol. 38, pp. 1257-65, July 1990.

[2] Swami, A. and J.M. Mendel, "Identifiability of the AR parameters of an ARMA process using cumulants," *IEEE Trans. on Automatic Control*, Vol. 37, pp. 268-73, Feb. 1992.

Purpose

Estimates impulse response via lag-domain bicepstral method

Syntax

```
[hest,ceps,a,b,minh,maxh] = biceps(y,p,q)
[hest,ceps,a,b,minh,maxh] = biceps(y,p,q,samp_seg,
   . . . overlap,flag,lh)
```

Description

`biceps` estimates the impulse response of the linear process, `y`, using the bicepstrum (lag-domain) method.

Variables `p` and `q` denote the number of causal and anticausal cepstral parameters to be estimated. The total length of the complex cepstrum is $p + q + 1$.

Variables `samp_seg`, `overlap`, and `flag` control the manner in which sample cumulants are estimated:

`samp_seg` specifies the number of samples per segment. Its default value is the row dimension of `y`; if `y` is a row vector, the column dimension is used as the default value.

`overlap` specifies the percentage overlap between segments; the allowed range is [0,99]; and the default value is 0.

If `y` is a matrix, the columns are assumed to correspond to independent realizations or records; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension.

If `flag` is `'b'`, then biased sample estimates of cumulants are estimated (default); if the first letter is not 'b', unbiased estimates are computed; only the first letter of `flag` is checked. Cumulants are estimated from each record, and then averaged across the set of records.

`lh`: the default value is `2(p+q)`; the estimated impulse response will range from samples `–lh` to `lh`.

`hest` is the estimated impulse response, $h(n)$, $n = -$`lh`, . . . , `lh`; the impulse response is normalized so that $h(0) = 1$. Samples of $h(n)$, $n < 0$, will have significant values if the original linear system is not minimum-phase (e.g., the original system is an ARMA model, some of whose zeros or poles lie outside the unit circle).

`ceps` is the estimated complex cepstrum $\hat{h}(n)$, $n = -q, \ldots, p$.

`a` is the vector of the minimum-phase cepstral parameters, $A^{(n)}$, $n = 1, \ldots, p$.

`b` is the vector of the maximum-phase cepstral parameters, $B^{(n)}$, $n = 1, \ldots, q$.

`minh` is the minimum-phase component of the IR;
`maxh` is the maximum-phase component of the IR; so that
`hest = conv(minh,maxh)`.

The cepstral parameters and complex cepstral coefficients are related via

$$\hat{h}(n) = \begin{cases} -A^{(n)}/n & n > 0 \\ B^{(-n)}/n & n < 0 \end{cases}$$

Any linear system can be approximated by an MA(L) model, provided L is large enough. If the MA model has Li zeros inside the unit circle, and $L_o = L - Li$ zeros outside the unit circle, then, the estimated impulse response will show an apparent shift of $L_o$samples to the left of time zero.

Algorithm

Details of the algorithm are given in the “Tutorial”.

See Also

`bicepsf`

Reference

[1] Pan, R. and C.L. Nikias, “The complex cepstrum of higher-order cumulants and non-minimum phase system identification,” *IEEE Trans ASSP*, Vol. 36, pp. 186-205, Feb. 1988.

Purpose

Estimates impulse response via frequency-domain bicepstral method

Syntax

```
[hest,ceps] = bicepsf(y,nlag,samp_seg,overlap,flag,nfft,wind)
```

Description

`bicepsf` estimates the impulse response of the linear process, `y`, using the bicepstrum Fast Fourier Transform (FFT) method.

`y` is the data vector or matrix.

`nlag` specifies the number of cumulant lags to be computed; the third-order cumulants of `y`, $C_{3y}(m,n)$,will be estimated for
`-nlag <=` m,n `<= nlag`. This parameter must be specified. A useful rule of thumb is to set `nlag` to `nsamp`/10, where `nsamp` is the length of the time series y.

Variables `samp_seg`, `overlap`, and `flag` control the manner in which sample cumulants are estimated:

`samp_seg` specifies the number of samples per segment or record. Its default value is the row dimension of `y`; if `y` is a row vector, the column dimension is used as the default value.

`overlap` specifies the percentage overlap between segments. The default value is 0. The allowed range is [0,99].

If `y` is a matrix, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension.

If `flag` is `'b'`, then biased sample estimates of cumulants are estimated (`default`); if the first letter is not 'b', unbiased estimates are computed; only the first letter of `flag` is checked.

Cumulants are estimated from each record, and then averaged across the set of records.

`nfft`: specifies the FFT size to be used for computing the bispectrum and the bicepstrum; longer FFT lengths yield better estimates, but also require more storage (two arrays of size `nfft` by `nfft`). The default value is 128; if `nfft` is smaller than `2*nlag+1`, the power of 2 just larger than `2*nlag+1` will be used.

`wind` specifies the lag-domain smoothing window. If `wind` is 0, the Parzen window will be applied. Otherwise, the usual unit hexagonal window will be applied. The Parzen window is the default.

The 1-D Parzen window is defined by,

$$d_p(m) = \begin{cases} 1 - 6\left(\frac{|m|}{L}\right)^2 + 6\left(\frac{|m|}{L}\right)^3 & |m| \le L/2 \\ 2\left(1 - \frac{|m|}{L}\right)^3 & L/2 \le |m| \le L \\ 0 & |m| > L \end{cases}$$

where L = `nlag`. The actual window applied to the estimated cumulants is given by,

$$W(m, n) = d_p(m)d_p(n)d_p(m - n).$$

The unit hexagonal window is given by,

$$W(m, n) = d(m)d(n)d(m - n)$$

where $d(m) = 1$, $|m| \le$ `nlag`.

`hest` is the estimated impulse response, $h(n)$, $n = -nfft/2, \ldots, nfft/2 - 1$. Samples of $h(n)$, $n < 0$, will have significant values if the original linear system is not minimum-phase (e.g., the original system is an ARMA model, some of whose zeros or poles lie outside the unit circle).

`ceps` is the estimated complex cepstrum, $\hat{h}(n)$, $n = -nfft/2, \ldots, nfft/2 - 1$.

Note that the method has an inherent scale and shift ambiguity. Any linear system can be approximated by an MA(L) model, provided L is large enough. If the MA model has Li zeros inside the unit circle, and $L_o = L - L_i$ zeros outside the unit circle, then, the estimated impulse response will show an apparent shift of L_o samples to the left of time zero.

Algorithm

Details of the algorithm are given in the “Tutorial”.

See Also

`biceps`

Reference

[1] Pan, R. and C.L. Nikias, “The complex cepstrum of higher-order cumulants and non-minimum phase system identification,” *IEEE Trans ASSP*, Vol. 36, pp. 186-205, Feb. 1988.

Purpose

Bicoherence estimation using the direct (FFT-based) method

Syntax

```
[bic, waxis] = bicoher(y)
[bic, waxis] = bicoher(y,nfft,wind,samp_seg,overlap)
```

Description

The bicoherence of the process y is estimated via the direct (FFT-based) method.

y is the data vector or matrix.

nfft specifies the FFT length to use for computing the bicoherence; the nominal default value is 128; if nfft is smaller than samp_seg, the power of 2 just larger than samp_seg will be used.

wind specifies the *time-domain* window to be used; it should be a vector of length seg_samp. By default, the hanning window is used. Data segments are multiplied by the time-domain window, then Fourier transformed to compute the frequency-domain double and triple products. The Fourier transform of the window function should be real and nonnegative.

samp_seg specifies the number of samples per segment or record. The default value is set such that eight (possibly overlapped) records are obtained.

overlap specifies the percentage overlap between segments. The default value is 0. The allowed range is [0,99].

If y is a matrix, the columns are assumed to correspond to independent realizations or records; in this case overlap is set to zero, and samp_seg is set to the row dimension.

bic is the estimated bicoherence; it is an nfft-by-nfft array, with origin at the center, and axes pointing down and to the right.

waxis is the set of frequencies associated with the bicoherence in bic. Thus, the *i*th row (or column) of bic corresponds to the frequency waxis(i), i=1,. . .,nfft. Frequencies are normalized; that is, the sampling frequency is assumed to be unity.

A contour plot of the magnitude of the estimated bicoherence is displayed.

Algorithm

The data, y, are segmented into possibly overlapping records; the mean is removed from each record, the time-domain window is applied, and the FFT computed; the bispectrum of the *k*th record is computed as,

$B_k(m, n) = X_k(m)X_k(n)X_k^*(m + n)$, where X_k denotes the FFT of the kth record, and the spectrum is computed as $P_k(m) = |X_k(m)|^2$. The spectral and bispectral estimates are averaged across records, and the bicoherence is then estimated as

$$\text{bic}(f_1, f_2) = \frac{\left|B(f_1, f_2)\right|^2}{P(f_1)P(f_2)P(f_1 + f_2)}$$

where $B(f_1,f_2)$ is the final estimate of the bispectrum, and $P(f)$ is the final estimate of the power spectrum.

See Also

`bispecd`, `bispeci`

References

[1] Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

[2] Nikias, C.L. and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

Purpose

Cross-bicoherence estimation using the direct (FFT-based) method

Syntax

```
[bicx, waxis] = bicoherx(w,x,y)
[bicx, waxis] = bicoherx(w,x,y,nfft,wind,samp_seg,overlap)
```

Description

The cross-bicoherence of the three processes `w`, `x` and `y` is estimated via the direct (FFT-based) method.

`w`,`x`,`y` should all have the same dimensions.

`nfft` specifies the FFT length to be used for computing the cross-bicoherence; the nominal default value is 128; if `nfft` is smaller than `samp_seg`, the power of 2 just larger than `samp_seg` will be used.

`wind` specifies the time-domain window to be used; it should be a vector of length `samp_seg`. By default, the `hanning` window is used. Data segments are multiplied by the time-domain window, then Fourier transformed to compute the frequency-domain double and triple products. The Fourier transform of the window function should be real and nonnegative.

`samp_seg` specifies the number of samples per segment. The default value is set such that eight (possibly overlapped) records are obtained.

`overlap` specifies the percentage overlap between segments. The default value is 50. The allowed range is [0,99].

If `w`,`x`,`y` are matrices, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension.

`bicx` is the estimated cross-bicoherence; it is an `nfft`-by-`nfft` array, with origin at the center, and axes pointing down and to the right.

`waxis` is the set of frequencies associated with the cross-bicoherence in `bicx`. Thus, the *i*th row (or column) of `bicx` corresponds to the frequency `waxis(i)`, `i=1,. . .,nfft`. Frequencies are normalized; that is, the sampling frequency is assumed to be unity.

A contour plot of the magnitude of the estimated cross-bicoherence is displayed.

Algorithm

The data, w,x,y, are segmented into possibly overlapping records; the mean is removed from each record, the time-domain window is applied, and the FFT computed; the cross-bispectrum of the kth record is computed as, $B_{wxy,k}(m,n) = W_k(m)X_k(n)Y_k^*(m+n)$, where W_k, X_k, and Y_k denote the FFT of the kth segments of w, x, and y. The spectra are computed as $P_{w,k}(m) = |W_k(m)|^2$, $P_{x,k}(m) = |\mathrm{Xl}(m)|^2$, and $Py,k(m) = |Y_k(m)|^2$. The spectral and cross-bispectral estimates are averaged across records, and the cross-bicoherence is then estimated as

$$\mathrm{bic}(f_1, f_2) = \frac{\left|B_{wxy}(f_1, f_2)\right|^2}{P_w(f_1)P_x(f_2)P_y(f_1+f_2)}$$

where $B_{wxy}(f_1,f_2)$ is the averaged estimate of the cross-bispectrum, and $P_w(f)$, $P_x(f)$, and $P_y(f)$ are the averaged estimates of the power spectra of w,x and y.

See Also

bicoher, bispecdx

References

[1] Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

[2] Nikias, C.L. and A. Petropulu, *Higher-Order Spectra Analysis: A Nonlinear Signal Processing Framework*, New Jersey: Prentice-Hall, 1993.

Purpose

Bispectrum estimation using the direct (FFT-based) method

Syntax

```
[bspec, waxis] = bispecd(y)
[bspec, waxis] = bispecd(y,nfft,wind,samp_seg,overlap)
```

Description

The bispectrum of the process `y` is estimated via the direct (FFT-based) method.

`y` is the data vector or matrix.

`nfft` specifies the FFT length to be used for computing the bispectrum; the nominal default value is 128; if `nfft` is smaller than `samp_seg`, the power of 2 just larger than `samp_seg` will be used.

`wind` specifies the frequency-domain smoothing window.

If `wind` is a scalar, the Rao-Gabr window of length `wind` will be used. This window is defined by [2],

$$W(m,n) = \frac{\sqrt{3}}{\pi^3}\left[1 - \frac{m^2 + n^2 + mn}{N^2}\right], \qquad (m,n) \in G$$

where N is half the FFT length, `nfft`, and G is the set of points, (m,n), satisfying,

$$m^2 + n^2 + mn \le \frac{\text{wind}^2}{(\text{nfft}/2)^2}.$$

- A unity value for `wind` results in no windowing.
- If `wind` <= 0, the default value of 5 will be used.
- If `wind` is a vector, it is assumed to specify a 1-D window from which a 2-D window is computed, $W(m,n) = w(m)w(n)w(m+n)$[1]-[2].
- If `wind` is a 2-D matrix, it is assumed to specify the 2-D smoother directly. The bispectrum estimate averaged across records is smoothed by convolving with the 2-D window function. The window function should be real and nonnegative.

`samp_seg` specifies the number of samples per segment. The default value is set such that eight (possibly overlapped) records are obtained.

overlap specifies the percentage overlap between segments. The default value is 50. The allowed range is [0,99].

If y is a matrix, the columns are assumed to correspond to independent realizations; in this case overlap is set to zero, and samp_seg is set to the row dimension.

bspec is the estimated bispectrum; it is an nfft-by-nfft array, with origin at the center, and axes pointing down and to the right.

waxis is the set of frequencies associated with the bispectrum in bspec. Thus, the *i*th row (or column) of bspec corresponds to the frequency waxis(i), i=1,. . .,nfft. Frequencies are normalized; that is, the sampling frequency is assumed to be unity.

A contour plot of the magnitude of the estimated bispectrum is displayed.

Algorithm

The data, y, are segmented into possibly overlapping records; the mean is removed from each record, and the FFT computed; the bispectrum of the *k*th record is computed as, $B_k(m, n) = X_k(m)X_k(n)X_k^*(m + n)$, where X_k denotes the FFT of the *k*th record, where denotes the FFT of the *k*th record. The bispectral estimates are averaged across records, and an optional frequency-domain smoother (specified by parameter wind) is applied.

See Also

bispeci

References

[1] Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

[2] Nikias, C.L. and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

Purpose

Cross-bispectrum estimation using the direct (FFT) method

Syntax

```
[bspec,waxis]=bispecdx(x,y,z,nfft,wind,samp_seg,overlap,flag)
```

Description

The cross-bispectrum of the three processes, `x`, `y`, and `z`, $B_{xyz}(\omega_1,\omega_2)$, is estimated via the direct (FFT) method.

`x`, `y`, and `z` should have the same dimensions.

`nfft` specifies the FFT length to be used for computing the cross-bispectrum; the nominal default value is 128; the actual FFT size used will be `max(samp_seg, nfft)`.

`wind` specifies the frequency-domain smoothing window. If `wind` is a scalar, the Rao-Gabr window [2]

$$W(m,n) = \frac{\sqrt{3}}{\pi^3}\left[1 - \frac{m^2 + n^2 + mn}{N^2}\right], \qquad (m,n) \in G$$

of length `wind` will be used; here is half the FFT length, `nfft`, and is the set of points, (m,n), satisfying,

$$m^2 + n^2 + mn \le \frac{\text{wind}^2}{(\text{nfft}/2)^2}.$$

- A unity value for `wind` results in no windowing.
- If `wind` <=0, the default value of 5 will be used.
- If `wind` is a vector, it is assumed to specify a 1-D window from which a 2-D window is computed, $W(m,n) = w(m)w(n)w(m+n)$ [1]-[2].
- If `wind` is a 2-D matrix, it is assumed to specify the 2-D smoother directly. The bispectrum estimate averaged across records is smoothed by convolving with the 2-D window function.

`samp_seg` specifies the number of samples per segment. The default value is set such that eight (possibly overlapped) records are obtained.

`overlap` specifies the percentage overlap between segments. The default value is 50. The allowed range is [0,99].

If `y` is a matrix, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension.

`flag` – a contour plot of the estimated cross-bispectrum will be displayed only if `flag` is nonzero; the default value is 1.

`bspec` is the estimated cross-bispectrum. It is an `nfft`-by-`nfft` array, with origin at the center, and axes pointing down and to the right.

`waxis` is the set of frequencies associated with the cross-bispectrum in `bspec`; thus, the *i*th row (or column) of `bspec` corresponds to the frequency `waxis(i), i=1, . . ., nfft`. Frequencies are normalized; that is, the sampling frequency is assumed to be unity.

Algorithm

The cross-bispectrum definition used in this routine is given by,

$$B_{xyz}(\omega_1, \omega_2) := E\{X(\omega_1)Y(\omega_2)Z^*(\omega_1 + \omega_2)\},$$

and is the 2-D Fourier transform of the cross-cumulant defined by

$$C_{xyz}(m, n) := E\{x(t + m)y(t + n)z^*(t)\}.$$

For a complex process, the cross-cumulant may also be defined by conjugating one or more of the terms *x*, *y*, and *z*. This is readily accomplished by using the MATLAB function `conj`.

`x`, `y`, and `z` are segmented into possibly overlapping records; the mean is removed from each record, and the FFT computed. The cross-bispectrum of the *k*th record is computed as,

$$B_{xyz,\,k}(m, n) = X_k(m)Y_k(n)Z_k^*(m + n)$$

where X_k, Y_k and Z_k are the FFT's of the *k*th segments of *x*, *y*, and *z*. The bispectral estimates are averaged across records, and an optional frequency-domain smoother (specified by parameter `wind`) is applied.

See Also

`bispecd`

References

[1] Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

[2] Nikias, C.L. and A. Petropulu, *Higher-Order Spectra Analysis: A Nonlinear Signal Processing Framework*, New Jersey: Prentice-Hall, 1993.

Purpose

Bispectrum estimation using the indirect method

Syntax

```
[bspec,waxis] = bispeci(y,nlag)
[bspec,waxis] = bispeci(y,nlag,samp_seg,
                        overlap,flag,nfft,wind)
```

Description

The bispectrum of the process y is estimated via the indirect method.

y is the data vector or matrix.

nlag specifies the number of cumulant lags to be computed; the third-order cumulants of y, $C_{3y}(m,n)$, will be estimated for –nlag <= m, n <= nlag. This parameter must be specified. A useful rule of thumb is to set nlag to nsamp/10, where nsamp is the length of the time series y.

samp_seg specifies the number of samples per segment or record. The default value of samp_seg is the length of the time series.

overlap specifies the percentage overlap between segments. The default value is 0. The allowed range is [0,99].

If y is a matrix, the columns are assumed to correspond to independent realizations; in this case overlap is set to zero, and samp_seg is set to the row dimension.

flag should be either biased (unbiased) for biased(unbiased) sample estimates of cumulants. By default, biased estimates are computed.

If the first letter is not '*b*', unbiased estimates are computed.

nfft specifies the FFT length to be used for computing the bispectrum; the default value is 128; if nfft is smaller than 2*nlag+1, the power of 2 just larger than 2*nlag+1 will be used.

wind specifies the lag-domain smoothing window. If wind is 0, the Parzen window will be applied. Otherwise, the usual unit hexagonal window will be applied. The Parzen window is the default.

The 1-D Parzen window is defined by,

$$d_p(m) = \begin{cases} 1 - 6\left(\frac{|m|}{L}\right)^2 + 6\left(\frac{|m|}{L}\right)^3 & |m| \le L/2 \\ 2\left(1 - \frac{|m|}{L}\right)^3 & L/2 \le |m| \le L \\ 0 & |m| > L \end{cases}$$

where L = `nlag`. The actual window applied to the estimated cumulants is given by,

$$W(m,n) = d_p(m)d_p(n)d_p(m-n).$$

The unit hexagonal window is given by,

$$W(m,n) = d(m)\mathrm{d}(n)d(m-n).$$

where $d(m) = 1$, $|m| \le$ `nlag`.

`bspec` is the estimated bispectrum; it is an `nfft`-by-`nfft` array, with origin at the center, and axes pointing down and to the right.

`waxis` is the set of frequencies associated with the bispectrum in `bspec`. Thus, the *i*th row (or column) of `bspec` corresponds to the frequency `waxis(i)`. Frequencies are normalized; that is, the sampling frequency is assumed to be unity.

Algorithm

The data, `y`, are segmented into possibly overlapping records; biased or unbiased sample estimates of third-order cumulants are computed for each record and then averaged across records; a lag window is applied to the estimated cumulants, and the bispectrum is obtained as the 2-D FFT of the windowed cumulant function.

See Also

`bispecd`

References

[1] Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

[2] Nikias, C.L. and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

Purpose

Theoretical bispectrum of an ARMA process

Syntax

```
[bspec, waxis] = bispect(ma,ar,nfft)
```

Description

The theoretical bispectrum corresponding to an ARMA process is computed.

`ma` is the MA parameter vector, and must be specified.

`ar` is the AR parameter vector; its default value is [1.0].

`nfft` specifies the FFT length to be used for computing the bispectrum; the default value is 512.

`bspec` is the bispectrum corresponding to the ARMA model. It is an `nfft`-by-`nfft` array, with origin at the center, and axes pointing down and to the right.

`waxis` is the set of frequencies associated with the bispectrum in `bspec`; thus, the *i*th row (or column) of `bspec` corresponds to the frequency `waxis(i)`. The sampling frequency is assumed to be unity.

Algorithm

Let $H(f) = B(f)/A(f)$ denote the transfer function of the ARMA filter; then, the bispectrum is given by,

$$B(f_1,f_2) = H(f_1)H(f_2)H^*(f_1 + f_2).$$

See Also

`cumtrue`, `trispect`

References

[1] Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

[2] Nikias, C.L. and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

cum2x

Purpose

Computes the cross-cumulant (covariance) of two signals

Syntax

```
cvec = cum2x(x,y,maxlag,samp_seg,overlap,flag)
```

Description

Computes the second-order cross-cumulant (covariance) of the two signals, `x` and `y`.

`x`, `y` should have identical dimensions.

`maxlag` specifies the maximum lag of the cumulant to be computed; its default value is 0.

`samp_seg` specifies the number of samples per segment. Its default value is the row dimension of `y`; if `y` is a row vector, the column dimension is used as the default value.

`overlap` specifies the percentage overlap between segments; the allowed range is [0,99]; the default value is 0.

If `flag` is biased, then biased sample estimates are computed (default); if the first letter is not 'b', unbiased estimates are computed.

`cvec` will contain the sample estimates of

$E\{(x^*(n) - \mu_x^*)(y(n+m) - \mu_y)\}$, m = –`maxlag, . . . , maxlag`.

Here μ_x denotes the mean of process `x`, and the superscript * denotes complex conjugation.

If `x`, `y` are matrices, columns are assumed to correspond to different realizations; in this case, `overlap` is set to 0, and `samp_seg` to the row dimension; the cross-cumulant is estimated from each realization, and then averaged across the suite of realizations.

Algorithm

Let $x(n)$ and $y(n)$, $n = 1, \ldots, N$, denote the two time series. Let M denote the number of samples per segment (the value of the variable `samp_seg`). Let O denote the percentage overlap between segments (the value of the variable `overlap`). Let $M_1 = M - M * O/100$. Then, the time series, `x` and `y`, are segmented into K records of M samples each, where $K = (N - M * O/100)/M$. The kth record or segment of `x` consists of the samples

$$x_k(i) = x(i + (k - 1) * M_1),\ i = 1, \ldots, M;\ k = 1, \ldots, K;$$

$y_k(i)$ is defined similarly.

The sample mean is removed from each record, and sample estimates of the cross-covariance are computed as

$$C_k(m) = \frac{1}{M(m)} \sum_i x_k^*(i) y_k(i + m),$$

where the summation over i extends from $1 + \max(0, -m)$ to $M - \max(0, m)$. The normalizing parameter $M(m)$ is identically equal to M for biased estimates, and equals $M - \max(0, m) - \max(0, -m)$ for unbiased estimates. The cumulants estimated from the K records are then averaged to obtain the final estimate,

$$C(m) = \frac{1}{K} \sum_{k=1}^{K} C_k(m).$$

See Also

`cum3x`, `cum4x`, `cumest`

cum3x

Purpose

Computes the third-order cross-cumulant of three signals

Syntax

```
cvec = cum3x(x,y,z,maxlag,samp_seg,overlap,flag,k1)
```

Description

Computes the third-order cross-cumulant of the three signals, `x, y, and z`, which should have identical dimensions.

`maxlag` specifies the maximum lag of the cumulant to be computed; its default value is 0.

`samp_seg` specifies the number of samples per segment. Its default value is the row dimension of `y`; if `y` is a row vector, the column dimension is used as the default value.

`overlap` specifies the percentage overlap between segments; the allowed range is [0,99]; and the default value is 0.

If `flag` is biased, then biased sample estimates are computed (default); if the first letter is not `'b'`, unbiased estimates are computed.

Parameter `k1` controls which 1-D slice of the cross-cumulant is computed; the default value is 0. By varying `k1`, we can obtain the entire third-order cross cumulants.

`cvec` will contain the sample estimates of

$E\{(x^*(n) - \mu_x^*)(y(n+m) - \mu_y)(z(n+k1) - \mu_z\}$, $m = -$`maxlag, . . . , maxlag`.

Here μ_x denotes the mean of process x, and the superscript * denotes complex conjugation.

If `x, y, z` are matrices, columns are assumed to correspond to different realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension; the cross-cumulant is estimated from each realization, and then averaged across the suite of realizations.

Algorithm

Let $x(n)$, $y(n)$, and $z(n)$, $n = 1, \ldots, N$ denote the three time series. Let M denote the number of samples per segment (the value of the variable samp_seg). Let O denote the percentage overlap between segments (the value of the variable overlap). Let $M_1 = M - M * O/100$. Then, the time series, x, y, and z, are segmented into K records of M samples each, where $K = (N - M * O/100)/M$. The kth record or segment of x consists of the samples

$$x_k(i) = x(i + (k - 1) * M_1),\ i = 1, \ldots, M;\ k = 1, \ldots, K;$$

$y_k(i)$ and $z_k(i)$ are defined similarly.

The sample mean is removed from each record, and sample estimates of the third-order cross-cumulants are obtained as,

$$C_k(m, n) = \frac{1}{M(m, n)} \sum_i x_k^*(i) y_k(i + m) z_k(i + n),$$

where the summation over i extends from $1 + \max(0,-m,-n)$ to $M - \max(0, m,n)$. The normalizing parameter $M(m,n)$ is identically equal to M for biased estimates, and equals $M - \max(0,m,n) - \max(0,-m,-n)$ for unbiased estimates. The cumulants estimated from the K records are then averaged to obtain the final estimate,

$$C(m, n) = \frac{1}{K} \sum_{k=1}^{K} C_k(m, n).$$

See Also

cum2x, cum4x, cumest

cum4x

Purpose

Computes the fourth-order cross-cumulant of four signals

Syntax

```
cvec = cum4x(w,x,y,z,maxlag,samp_seg,overlap,flag,k1,k2)
```

Description

Computes the fourth-order cross-cumulant of the four signals, `w`, `x`, `y`, and `z`, which should have identical dimensions.

`maxlag` specifies the maximum lag of the cumulant to be computed; its default value is 0.

`samp_seg` specifies the number of samples per segment. Its default value is the row dimension of `y`; if `y` is a row vector, the column dimension is used as the default value.

`overlap` specifies the percentage overlap between segments; the allowed range is [0,99]; and the default value is 0.

If `flag` is biased, then biased sample estimates are computed (default); if the first letter is not `'b'`, unbiased estimates are computed.

Parameters `k1` and `k2` control which 1-D slice of the cross-cumulant is computed; the default value for both the parameters is 0. By varying `k1` and `k2`, we can obtain the entire fourth-order cross-cumulant.

`cvec` will contain the sample estimates of

$$\text{cum}(w^*(n),\, x(n+m),\, y(n+k1),\, z^*(n+k2)),\quad m = -\texttt{maxlag}, \ldots, \texttt{maxlag},$$

where the superscript * denotes complex conjugation, and $\text{cum}(a,b,c,d) = E(abcd) - E(ab)E(cd) - E(ac)E(bd) - E(ad)E(bc)$, and it is assumed that a,b,c,d are zero-mean random variables.

If `w`, `x`, `y`, `z` are matrices, columns are assumed to correspond to different realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension; the cross-cumulant is estimated from each realization, and then averaged across the suite of realizations.

Algorithm

Let $w(n)$, $x(n)$, $y(n)$ and $z(n)$, $n = 1, \ldots, N$ denote the three time series. Let M denote the number of samples per segment (the value of the variable samp_seg). Let O denote the percentage overlap between segments (the value of the variable overlap). Let $M_1 = M - M * O/100$. Then, the time series, w, x, y, and z, are segmented into K records of M samples each, where $K = (N - M * O/100)/M$. The kth record or segment of x consists of the samples

$$x_k(i) = x(i + (k-1) * M_1),\ i = 1, \ldots, M;\ k = 1, \ldots, K;$$

w_k(i), $y_k(i)$ and $z_k(i)$ are defined similarly.

The sample mean is removed from each record, and sample estimates of the fourth-order cross-cumulants are obtained as,

$$\begin{aligned} C_k(m,n,t) &= \frac{1}{M(m,n,t)}\sum_i w_k^*(i)x_k(i+m)y_k(i+n)z_k^*(i+t) \\ &- \left\{\frac{1}{M(m)}\sum_i w_k^*(i)x_k(i+m)\right\}\left\{\frac{1}{M(t-n)}\sum_i y_k(i)z_k^*(i+t-n)\right\} \\ &- \left\{\frac{1}{M(n)}\sum_i w_k^*(i)y_k(i+n)\right\}\left\{\frac{1}{M(t-m)}\sum_i x_k(i)z_k^*(i+t-m)\right\} \\ &- \left\{\frac{1}{M(t)}\sum_i w_k^*(i)z_k^*(i+t)\right\}\left\{\frac{1}{M(n-m)}\sum_i x_k(i)y_k(i+n-m)\right\} \end{aligned}$$

where the summation over i is such that the indices of $w(\cdot)$, $x(\cdot)$, $y(\cdot)$, and $z(\cdot)$ are all in the range [1,M]. The normalizing parameter $M(m,n,t)$ is identically equal to M for biased estimates, and equals $M - \max(0,m,n,t) - \max(0,-m,-n,-t)$ for unbiased estimates, and the normalizing parameter $M(m)$ is identically equal to M for biased estimates, and equals $M - \max(0,m) - \max(0,-m)$ for unbiased estimates. The cumulants estimated from the K records are then averaged to obtain the final estimate,

$$C(m,n,t) = \frac{1}{K}\sum_{k=1}^{K} C_k(m,n,t).$$

See Also

cum2x, cum3x, cumest

cumest

Purpose

Computes sample estimates of cumulants using the overlapped-segment method

Syntax

```
cvec = cumest(y)
cvec = cumest(y,norder,maxlag,samp_seg,overlap,flag,k1,k2)
```

Description

`cumest` computes sample estimates of a 1-D slice of the cumulants of the process `y`.

`y` is the data matrix or vector.

`norder` specifies the cumulant order, and should be 2, 3, or 4; the default value is 2.

`maxlag` specifies the maximum lag of the cumulant to be computed; its default value is 0.

`samp_seg` specifies the number of samples per segment; the default value is the length of the time series.

`overlap` specifies the percentage overlap between segments; maximum allowed value is 99; default value is 0.

If `flag` is biased, then biased sample estimates are computed (default); if the first letter is not `'b'`, unbiased estimates are computed.

If `y` is a matrix, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension. Cumulants are estimated from each realization, and then averaged.

Parameters `k1` and `k2` control which 1-D slice of the cumulant function is computed; their default values are zero.

`cvec` will contain $C_2(m)$, $C_3(m,k1)$ or $C_4(m,k1,k2)$,
m = –`maxlag, . . . , maxlag`, depending upon the specified cumulant order.

Note that `cumest` estimates a 1-D slice of the cumulant.

Algorithm

Let $y(n)$, $n = 1, \ldots, N$ denote the three time series. Let M denote the number of samples per segment (the value of the variable samp_seg). Let O denote the percentage overlap between segments (variable overlap). Let $M_1 = M - M * O/100$. Then, the time series, y, is segmented into K records of M samples each, where $K = (N - M * O/100)/M$. The kth record or segment of x consists of the samples

$$y_k(i) = y(i + (k-1) * M1),\ i = 1, \ldots, M;\ k = 1, \ldots, K.$$

The sample mean is removed from the kth record, and sample estimates of the cumulants are computed. For example, sample estimates of third-order cumulants are obtained as,

$$C_{3y,k}(m,n) = \frac{1}{M(m,n)} \sum_{i=1+\max(0,-m,-n)}^{M-\max(0,m,n)} y_k^*(i) y_k(i+m) y_k(i+n)$$

where $M(m,n) \equiv M$ for biased estimates, and $M - \max(0,m,n) + \min(0,m,n)$ for unbiased estimates. The final estimate is given by Second- and fourth-order

$$C_{3y}(m,n) = \frac{1}{K} \sum_{k=1}^{K} C_{3y,k}(m,n).$$

cumulant estimates are obtained similarly; see also the algorithm descriptions for M-files cum2x, and and cum4x.

See Also

cumtrue, cum2x, cum3x, cum4x

Reference

[1] Nikias, C.L. and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

cumtrue

Purpose

Computes theoretical (that is, true) cumulants of a linear process

Syntax

```
cmat = cumtrue(ma)
cmat = cumtrue(ma,ar,norder,nlags,k)
```

Description

`cumtrue` computes the theoretical (that is, true) cumulants of a linear (ARMA) process.

`ma` is the MA parameter vector, and must be specified. If `q` is the MA order, then `ma` will be of length `q+1`.

`ar` is the AR parameter vector; its default value is [1]. If `p` is the AR order, then `ar` will be of length `p+1`.

`norder` is the cumulant order; allowed values are 2, 3 and 4; the default value is 3.

`nlags` is the maximum number of cumulant lags to be computed. The default value of `nlags` is `q+p`.

If fourth-order cumulants are requested, that is, `norder` is 4, then, `k` refers to the third-lag of the fourth-order cumulant $C_4(i,j,k)$. Its default value is 0.

`cmat` is the vector or matrix of theoretical cumulants. If `norder` is 2, `cmat` is a column vector of length `2*nlags + 1`, and consists of $C_2(m)$, m = `nlags`,. . ., `nlags`.

If `norder` is 3 or 4, `cmat` is a `2*nlags + 1` by `2*nlags + 1` matrix. If `norder` is 3, the (i,j) element of the matrix is $C_3(i - \texttt{nlags} - 1, j - \texttt{nlags} - 1)$, and, if `norder` is 4, the (i,j) element of the matrix is $C_4(i - \texttt{nlags} - 1, j - \texttt{nlags} - 1, k)$. Note that the (`nlags+1`,`nlags+1`) element of the matrix `cmat` contains $C_3(0,0)$ or $C_4(0,0,k)$.

Algorithm

Let $h(n)$ denote the impulse response of a linear system, excited by an i.i.d. process, $u(n)$, with kth order cumulant, γ_{ku}. Then, the kth order cumulant of the output of the linear system is given by the Bartlett-Brillinger-Rosenblatt formula,

$$C_{ky}(\tau_1, \ldots, \tau_{k-1}) = \gamma_{ku} \sum_{n=0}^{\infty} h(n)h(n+\tau_1)\ldots h(n+\tau_{k-1})$$

In this module, γ_{ku} is assumed to be unity.

See Also

`bispect`, `trispect`

Reference

[1] Mendel, J.M., "Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications," *Proc. IEEE*, Vol. 79, pp. 278-305, 1991.

doa

Purpose

Direction of arrival estimation, based on spatial covariance or fourth-order cumulant matrix

Syntax

```
[spec,theta,dvec] = doa(ymat)
[spec,theta,dvec] = doa(ymat,dspace,dtheta,nsource,order,delta)
```

Description

doa estimates the angular spectra (of source bearings), for a uniformly spaced linear array, using the Eigenvector, Music, Pisarenko, ML (Capon), AR, minimum-norm, beamformer, and ESPRIT methods based either on fourth-order cumulants or the spatial covariance matrix. The peaks in the angular spectra nominally indicate the direction of arrival (DOA).

various algorithms based on the diagonal slice of the fourth-order cumulant or the spatial cross-correlation.

ymat is the data array; each column corresponds to a different sensor; the rows correspond to "snapshots."

dspace is the element spacing in wavelength units; the default value is 0.5 (half wavelength spacing)

dtheta is the angular spacing (in degrees) at which the "spectra" are to be computed; the default value is 2 degrees.

nsource is the number of sources; the default value is 0. If nsource is not positive you are prompted to choose the number of sources. This value can be inferred from the display of the singular values of the covariance or the fourth-order cumulant matrix as follows. If the singular values, $\sigma(k)$, are more or less constant for $k > p$, then, a useful rule-of-thumb is to choose p as the number of sources. (With finite data records, you can expect a slow decrease in the smaller singular values.) Usually, there will be a significant drop in the singular value from $\sigma(p)$ to $\sigma(p + 1)$.

order specifies the cumulant order to use; it should be either 2 (for cross-correlation based estimates) or 4 (for estimates based on the diagonal slice of the fourth-order cross-cumulant). The default value is 4.

delta is the displacement between the two subarrays for ESPRIT; here, we assume that the two subarrays are obtained by partitioning a single array. If there are m sensors (the column dimension of array ymat), then, the first array

will consist of sensors `1,2, . . .,m-delta`, and the second array will consist of sensors `delta, delta+1, . . ., m`. The default value of `delta` is 1.

`spec` is the array of estimated "spectra"; the columns correspond to estimates based on the Eigenvector, Music, Pisarenko, ML (Capon), AR, minimum-norm and beamformer methods; the rows correspond to bearing angles, which are returned in the vector `theta`. All estimated spectra are normalized to a maximum value of unity.

`theta` is the vector of bearings, corresponding to the rows of spec.

`dvec` is the vector of bearings estimated by ESPRIT.

Algorithm

Details of the algorithm are given in the "Tutorial".

See Also

`harmest`

References

[1] Johnson, D.H., "The application of spectrum estimation methods to bearing estimation problems," *Proc. IEEE*, Vol. 70, pp. 975-89, 1982.

[2] Pan, R. and C.L. Nikias, "Harmonic decomposition methods in cumulant domains," *Proc. ICASSP-88*, pp. 2356-59, New York, 1988.

[3] Roy, R. and T. Kailath, "ESPRIT – Estimation of signal parameters via rotational invariance techniques," *IEEE Trans ASSP*, Vol. 37, pp. 984-95, July 1989.

[4] Swami, A. and J.M. Mendel, "Cumulant-based approach to the harmonic retrieval and related problems," *IEEE Trans. ASSP*, Vol. 39, pp. 1099-1109, May 1991.

doagen

Purpose

Generates synthetics for the direction of arrival (DOA) problem, using a uniform linear array

Syntax

```
ymat = doagen
ymat = doagen(default)
```

Description

doagen generates synthetics for the DOA problem. The sensor array is assumed to be linear and equispaced (uniform).

zmat returns the generated data: each column corresponds to a different realization.

If doagen is invoked without any input arguments, then you are prompted for all parameters: the source bearings, number of sensors (*msens*), sensor spacing, number of samples per sensor record (*nsamp*), pdf of source signal, variance of additive noise, pdf of additive noise, and ARMA parameters for the noise spectrum. The sensor array is assumed to be linear and uniformly spaced.

If the user-specified noise variance is greater than zero, the sensor signals are standardized to unity variance, before the additive noise is added to them.

If the function is invoked as doagen(default), where the variable default may take on *any* value(s), then, the default settings are used. Matrix ymat in file doa1.mat was generated via ymat = doagen(1);.

ymat is an nsamp-by-msens array, where nsamp is the number of samples per sensor record, and msens is the number of sensors. The *k*th column contains the signal observed at the *k*th sensor.

Purpose

Computes decision statistics for Hinich's Gaussianity and linearity tests

Syntax

```
[sg, sl] = glstat(y)
[sg, sl] = glstat(y,cparm,nfft)
```

Description

`glstat` estimates the decision statistics for Hinich's Gaussianity and linearity tests.

The bispectrum is estimated using the direct method, and a frequency-domain 2-D boxcar smoother is applied. The power spectrum is estimated via the direct method, and a boxcar smoother is applied. The bicoherence is then estimated. The Gaussianity test (actually zero-skewness test) basically involves deciding whether or not the estimated bicoherence is zero. The linearity test involves deciding whether or not the estimated bicoherence is constant.

`y` is the time series (should be a vector).

`cparm` is the resolution parameter; it should lie between 0.5 and 1.0, if a single record is used; the default value is 0.51. Increasing `cparm` decreases the variance of the smoothed bispectral and spectral estimates, but at the expense of poorer resolution.

`nfft` is the FFT length to be used; the default length is 128. If the length of `y` is greater than `nfft`, `y` is segmented into records of length `nfft`.

The boxcar window length, `M`, is the value obtained by rounding off $(\texttt{nfft})^{\texttt{cparm}}$.

`sg`, the statistic for the Gaussianity test, is a three-element vector:

`sg(1)` is the estimated statistic `S`.

`sg(2)` is the number of degrees of freedom (df), `p`.

`sg(3)` is the probability of false alarm (`Pfa`).

More specifically, it is the probability that a χ^2 random variable with `p` degrees of freedom could have a value larger than the estimated `S` in `sg(1)`. If this probability is small, say, 0.05, then we may reject the hypothesis of zero skewness at a `Pfa` (or significance level) of 0.05. In other words, if you decide to accept the hypothesis that the data have nonzero skewness, then the probability that the data may actually have zero skewness is given by `sg(3)`. If `Pfa` is large, then the hypothesis of zero skewness cannot be easily rejected.

`sl`, the statistic for the linearity test, is a three-element vector: (see below for more details):
`sl(1)` is the estimated statistic `R`.
`sl(2)` is the estimated parameter λ (this parameter is called λ_o in Hinich's paper).
`sl(3)` is the theoretical value of `R`.

The linearity hypothesis should be rejected if the estimated statistic, `R`, is much larger or much smaller than the interquartile range of $\chi_2^2(\lambda)$ (the χ^2 distributed random variable, with 2 degrees of freedom, and noncentrality parameter, λ) [1]. In practice, for a nonlinear process, the estimated `R` value may be expected to be much larger than the theoretical `R` value [1]. The number of samples available to estimate the interquartile range is also printed; note that the estimate cannot be reliable if the number of samples is small.

Algorithm

Let $B_{3y}(\omega_1,\omega_2)$ denote the bispectrum, and let $P_{yy}(\omega)$ denote the power spectrum. The normalized bispectrum (or bicoherence) is defined as

$$\mathrm{bic}_y(\omega_1, \omega_2) = \frac{B_{3y}(\omega_1, \omega_2)}{(P_{yy}(\omega_1)P_{yy}(\omega_2)P_{yy}(\omega_1 + \omega_2))^{1/2}}.$$

Under the Gaussianity (zero skewness) assumption, the expected value of the bicoherence is zero, that is, $E\{\mathrm{bic}_y(\omega_1,\omega_2)\} = 0$. The test of Gaussianity is based on the mean bicoherence power,

$$S = \sum \left|\mathrm{bic}_y(\omega_1, \omega_2)\right|^2,$$

where the summation is performed over the nonredundant region of the bispectrum; details are given in [1]. The statistic `S` is χ^2 distributed, with `p` degrees of freedom, where `p` is a function of the FFT length, `nfft`, and the resolution parameter, `cparm` [1].

Under the linearity assumption, $B_n(\omega_1,\omega_2)$ is constant for all ω_1 and ω_2.

Let

$$X(\omega_1, \omega_2) = \frac{1}{\sqrt{N^{1-2*\mathrm{cparm}}}} B_n(\omega_1, \omega_2).$$

An estimate of $\lambda := (2N^{2*\texttt{cparm}-1})\gamma_{3x}$ is obtained. Note that $X(\omega_1,\omega_2)$ is chi-squared distributed with noncentrality parameter λ. The sample interquartile range `R` of the $X(m,n)$'s is estimated, and should be compared with the theoretical interquartile range of a chi-squared distribution with two degrees of freedom and noncentrality parameter λ.

These statistics are defined in Hinich's paper [1]. Sankaran's approximations in [2] are used to estimate the `Pfa` and the theoretical interquartile value of $\chi_2^2(\lambda)$.

References

[1] Hinich, M.J., "Testing for Gaussianity and linearity of a stationary time series," *J. Time Series Analysis*, Vol. 3, pp. 169-76, 1982.

[2] Patel, J.K. and C.B. Read, *Handbook of the Normal Distribution*, Sec 7.9.4, M. Dekker, New York, 1982.

harmest

Purpose

Estimation of frequencies of harmonics in colored Gaussian noise, and *power spectra*

Syntax

```
[Pxx,ar1,ar2] = harmest(y)
[Pxx,ar1,ar2] = harmest(y,maxlag,p_order,flag,nfft,norder)
```

Description

`harmest` estimates the frequencies of real harmonics in noise, and power spectra using the MUSIC, Eigenvector, Pisarenko, ML (Capon), AR and minimum-norm methods based either on the diagonal slice of fourth-order cumulants, or on the covariance; it also estimates the conventional periodogram.

`y` is the data matrix; each of its columns is assumed to correspond to a different realization.

`maxlag` specifies the number of cumulant lags to be computed;

`maxlag` should be greater than twice the maximum number of harmonics expected. The default value of `maxlag` is arbitrarily set to `nsamp`/12, where `nsamp` is the row dimension of the data matrix.

`p_order` specifies the order (must be greater than or equal to twice the number of harmonics); if this parameter is not specified, or is not positive, you are prompted for the value of `p_order`. The order can be inferred from the display of the the singular values of the cumulant matrix as follows. If the singular values, $\sigma(k)$, are more or less constant for $k > p$, then, a useful rule-of-thumb is to choose p as the number of harmonics. (With finite data records, one should expect a slow decrease in the singular values.) Usually, there will be a significant drop in the singular value from $\sigma(p)$ to $\sigma(p + 1)$.

If `flag` is biased, biased sample estimates of cumulants are computed (`default`); if the first letter is not `'b'`, unbiased estimates are computed.

`nfft` specifies the FFT length; its default value is 256.

`norder` specifies the cumulant order to use; it should be either 2 (for covariance based estimates) or 4 (for estimates based on the diagonal slice of the fourth-order cumulant). The default value is 4.

`Pxx` is an `nfft/2` $\times$ `7` matrix, whose first five columns, respectively contain estimates of "power spectra" obtained via the MUSIC, Eigenvector, Pisarenko, ML (Capon) and AR methods based on the diagonal slice of the fourth-order

cumulant or the covariance function. The sixth column contains the standard power spectrum estimate obtained via the periodogram method. The last column is the estimate obtained by applying the minimum norm algorithm. All estimated spectra are normalized to a maximum value of unity for purposes of display only. The estimated "power spectra" are displayed on the MATLAB graphics window.

`ar1` is the estimated parameter vector for the AR method.

`ar2` is the estimated parameter vector for the minimum-norm method.

Algorithm

Let C denote the `maxlag` by `maxlag` matrix, with entries, $C(i,j) = C4y(i - j,0,0)$ or $C(i,j) = C_{2y}(i - j)$. Also, let $C = VSV'$ denote the eigen decomposition, where S is the diagonal matrix of eigenvalues, $\lambda(k)$, and V is the matrix of eigenvectors, $\mathbf{v}_k$, $k = 1, \ldots,$ `maxlag`. Let

$$\mathbf{e}(\omega) := [1, \exp(-j\omega), \ldots, \exp(-j(\text{maxlag} - 1)\omega)]'$$

denote the FFT vector; and let p denote the chosen order (the parameter `p_order`). Then, the "power spectral" estimates are obtained as follows.

$$P(\omega) = \left(\sum_{k=p+1}^{M} w(k) \left| \mathbf{e}'(\omega)\mathbf{v}_k \right|^2 \right)$$

where,

$$w(k)) = \begin{cases} 1 & \text{MUSIC} \\ 1/\lambda(k) & \text{Eigenvector} \\ \delta(k-m) & \text{Pisarenko} \end{cases}$$

where $\delta(k)$ is the Kronecker delta function.

The AR power spectrum is obtained as follows: First, a rank approximation of the matrix C is obtained, as $\hat{C} = V\hat{S}V'$, where $\hat{S}$ is obtained from S by setting $\lambda(k) = 0$, $k = p + 1, \ldots, M$. The AR parameter vector is then obtained as the solution to $\hat{C}\mathbf{a} = \mathbf{0}$; the method in [5] is used, and the solution is forced to have unity modulus.

The ML(Capon) solution is given by,

$$P_{\mathrm{ML}}(\omega) = \left(\sum_{k=1}^{p} \frac{1}{\lambda(k)} \left| \mathbf{e}'(\omega)\mathbf{v}_k \right|^2 \right)^{-1}$$

Let V denote the matrix of eigenvectors corresponding to the p largest eigenvalues of R. Partition matrix V as,

$$V = \begin{bmatrix} v_1^T \\ V_r \end{bmatrix}.$$

The AR parameter vector for the minimum-norm solution is given by

$$\mathbf{a} = \begin{bmatrix} 1 \\ -V_r^* v_1 / (1 - v^H v) \end{bmatrix}.$$

The power spectrum is given by

$$S(\omega) = 1 / \left| \sum_{k=0}^{p} a(k) \exp(-jk\omega) \right|^2.$$

References

[1] Pan, R., and C.L. Nikias, "Harmonic decomposition methods incumulant domains," *Proc. IEEE* ICASSP-88, pp. 2356-59, 1988.

[2] Swami, A., and J.M. Mendel, "Cumulant-based approach to the harmonic retrieval and related problems," *IEEE Trans. ASSP*, Vol. 39, pp. 1099-1109, May 1991; see also *Proc. ICASSP-88*, pp. 2264-67, 1988.

[3] Cadzow, J.A., "Spectral Estimation: An Overdetermined Rational Model Equation Approach," *Proc. IEEE*, Vol. 70, pp. 907-38, 1982.

[4] Haykin, S., *Adaptive Filter Theory*, New Jersey: Prentice-Hall, pp. 464-70, 2nd ed., 1991.

[5] Kumaresan, R. and D.W. Tufts, "Estimating the angles of arrival of multiple plane waves," *IEEE Trans. AES*, Vol. 19, pp. 134-39, 1983.

Purpose

Generates harmonics in colored Gaussian noise

Syntax

```
zmat = harmgen
zmat = harmgen(default)
```

Description

harmgen generates independent realizations of the signal

$$y(n) = \sum_{k=1}^{p} a_k \cos(2\pi\lambda_k n + \phi_k) + g(n)$$

If harmgen is invoked without any input arguments, then you are prompted for the length of the realizations, the number of realizations, the number of harmonics (p), their frequencies (λ_k) and amplitudes (α_k), and the variance of the additive colored Gaussian noise, $g(n)$.

The additive colored Gaussian noise, $g(n)$, is generated by passing a white Gaussian noise sequence through an user-specified ARMA filter (you are prompted for these ARMA parameters). For each realization, the phases ϕ_k are chosen randomly from an uniform distribution. Note that noise-free realizations can be obtained by specifying a value of zero for the noise variance (when prompted).

If the function is invoked as harmgen(default), where the variable default may take on *any* value(s), then, the default settings are used (see Examples below).

Each column of zmat corresponds to a different realization.

The matrix zmat in the file harm.mat can be regenerated via zmat = harmgen(1);.

hosademo

Purpose

A guided tour of the Higher-Order Spectral Analysis Toolbox

Syntax

```
hosademo
```

Description

`hosademo` takes you on a guided tour of the Higher-Order Spectral Analysis Toolbox; `hosademo` also offers a brief introduction to the area of higher-order statistics.

Purpose

Gives a one-line synopsis for all M-files in the Higher-Order Spectral Analysis Toolbox

Syntax

```
hosahelp
```

Description

`hosahelp` gives a one-line synopsis for all the documented M-files in the Higher-Order Spectral Analysis Toolbox.

hprony

Purpose

Estimates the parameters of a complex transient signal modeled as the sum of complex exponentials with decaying amplitudes

Syntax

```
[a,theta,alpha,fr] = hprony(x,p)
```

Description

Models a complex transient signal as the sum of complex exponentials with decaying amplitudes,

$$x(n) = \sum_{k=1}^{p} a(k)e^{j\theta(k)}\exp(n(\alpha(k) + j2\pi f(k))),$$

where p is the order; the $a(k)$'s are amplitudes, $\theta(k)$'s are the initial phases, $\alpha(k)$'s are the damping factors, and $f(k)$'s are the frequencies.

`x` is the time series; it must be a vector.

`p` is the order; default value is $n/10$, where n is the length of x.

`a` is the vector of estimated amplitudes.

`theta` is the vector of estimated initial phases.

`alpha` is the vector of estimated damping factors.

`fr` is the vector of estimated frequencies.

The input time series, and the time series corresponding to the estimated parameters are plotted.

Algorithm

The Signal Processing Toolbox function `prony` is used to fit an ARMA(p,p-1) model to the transient signal; the MATLAB function `residue` is used to convert the ARMA parameters to the pole-residue form; these are then converted to the amplitude, phase, frequency, and damping factor terms.

Note that estimates of the amplitude and starting phase are sensitive to the presence of additive noise.

Reference

[1] Krauss, T., J.N. Little, and L. Shure, *MATLAB Signal Processing Toolbox User's Guide*, The MathWorks Inc., 1994.

Purpose

Computes instrumental variables. Used by `rivdl` and `rivtr`

Syntax

```
z = ivcal(y)
z = ivcal(y,morder,lambda)
```

Description

`ivcal` computes the instrumental variable corresponding to the correlation, or to the diagonal slice of third- or fourth-order cumulants.

`y` is the time series. If `y` is a matrix, each column is treated independently.

`morder` is the cumulant order: it should be less than 5; default value is 3.

`lambda` is the forgetting factor; $0 <$ `lambda` ≤ 1; the default value is 1.

`z` is the computed instrumental variable. $z(n)$ is obtained from $y(n)$ as follows:

morder	z(n)
q <= 0	y(n+q)
1	sign (y(n))
2	y(n)
3	$y^2(n)$
4	$y^3(n) - 3s(n) * y(n)$

where, for $0 < \lambda < 1$,

$$s(n) = s(n-1) + \lambda y^2(n), \quad s(1) = y^2(1).$$

and, for $\lambda = 1$,

$$s(n) = \frac{1}{n} \sum_{k=1}^{n} y^2(k)$$

Reference

[1] Swami, A., and J.M. Mendel, "Adaptive Cumulant-Based Estimation of ARMA Parameters," *Proc. Amer. Control Conf., ACC-88*, Atlanta, GA, 2114-19, June 1988.

maest

Purpose

Estimates MA parameters using the modified GM method

Syntax

```
bvec = maest(y,q)
bvec = maest(y,q,norder,samp_seg,overlap,flag)
```

Description

`maest` estimates the parameters of the MA(q) process, `y`, via the modified GM method, based on cumulants of order `norder` and the autocorrelation.

`y` is the data vector or matrix.

`q` is the MA order.

`norder` should be 3 or 4; the default value is 3.

Variables `samp_seg`, `overlap`, and `flag` control the manner in which sample cumulants are estimated:

`samp_seg` specifies the number of samples per segment; the default value is the length of the time series.

`overlap` specifies the percentage overlap between segments; maximum allowed value is 99; default value is 0.

If `flag` is `biased`, then biased sample estimates are computed; (`default`); if the first letter is not `'b'`, unbiased estimates are computed.

If `y` is a matrix, the columns are assumed to correspond to independent realizations; in this case `overlap` is set to zero, and `samp_seg` is set to the row dimension. Cumulants are estimated from each realization, and then averaged.

The estimated MA parameters are returned in the $q + 1$ element column vector `bvec`.

Algorithm

The GM method obtains the least-squares solution to the set of equations,

$$\sum_{k=0}^{q} \varepsilon b(k) C_3(n-k, n-k) - \sum_{k=1}^{q} b^2(k) R(n-k) = R(n),$$

where $n = -q, \ldots, 2q$, and `norder` = 3. For `norder` = 4, the equations are

$$\sum_{k=0}^{q} \varepsilon b(k) C_4(n-k, n-k, n-k) - \sum_{k=1}^{q} b^3(k) R(n-k) = R(n),$$

where $n = -q, \ldots, 2q$. Here, R, C_3, and C_4 are the second-, third- and fourth-order cumulants, respectively.

In order to handle additive white noise, equations involving $R(0)$ are eliminated, since the variance of the noise is unknown.

The modified GM method incorporates a fix, due to Tugnait [2], which appends the following set of equations to the preceding set of equations for

$$\varepsilon_3 R(n) - \sum_{k=1}^{q} b(k) C_3(k-n, q) = C_3(-n, q)$$

n = $-q, \ldots, q$, or,

$$\varepsilon_4 R(n) - \sum_{k=1}^{q} b(k) C_4(k-n, q, 0) = C_4(-n, q, 0)$$

depending upon the specified cumulant order. The least-squares solution to the combined system of equations is obtained. Here, $\varepsilon_3 = \gamma_{3u} b(q) / \sigma_u^2$, and $\varepsilon_4 = \gamma_{4u} b(q) / \sigma_u^2$.

When third-order cumulants are used, the method estimates both $b(k)$ and $b^2(k)$. Let $b_1(k)$ and $b_2(k)$ denote the estimates of $b(k)$ and $b^2(k)$. If all the estimated $b^2(k)$'s are nonnegative, the final MA parameter estimate is obtained as,

$$\hat{b}(k) = sign(b_1(k)) * \sqrt{0.5(b_1^2(k) + b_2(k))};$$

otherwise, $\hat{b}(k) = b_1(k)$.

When fourth-order cumulants are used, the method estimates both $b(k)$ and $b^3(k)$. Let $b_1(k)$ and $b_3(k)$ denote the estimates of $b(k)$ and $b^3(k)$. If all the

estimated $b^3(k)$'s have the same sign as the corresponding $b_1(k)$'s, then the final MA parameter estimate is obtained as,

$$\hat{b}(k) = sign(b_1(k))*(|b_1(k)| + |b_3(k)|^{1/3})/2;$$

otherwise, $\hat{b}(k) = b_1(k)$.

References

[1] Giannakis, G.B. and J.M. Mendel, "Identification of non-minimum phase systems using higher-order statistics," *IEEE Trans. ASSP*, Vol. 37, pp. 360-77, Mar. 1989.

[2] Tugnait, J.K., "Approaches to FIR system identification with noisy data using higher-order statistics," *IEEE Trans. ASSP*, Vol. 38, pp. 1307-17, July 1990.

Purpose

Estimates the order of an MA process

Syntax

```
q = maorder(y,qmin,qmax,pfa,flag)
```

Description

Estimates the order of an MA process using third-order cumulants.

`y` is the observed MA process and *must* be a vector.

`qmin` specifies the minimum expected MA order; the default value is 0.

`qmax` specifies the maximum expected MA order; the default value is 10.

`pfa` specifies the probability of false alarm for the hypthesis testing procedure; the default value is 0.05.

`flag` – if `flag` is nonzero, the sample estimate of $C_3(q,k)$, its estimated variance, the threshold corresponding to the specified `pfa`, and whether the absolute value of the estimated $C_3(q,k)$ is less than the threshold are displayed on the command window. The default value is 1.

`q` is the estimated MA order.

Algorithm

The basic idea is that for an MA(q) process, the true values of the cumulant $c_{3y}(m,0)$ will be identically zero, if m > q. When the true cumulants are replaced by sample estimates, the estimated values of $c_{3y}(q + 1,0)$, $i > 0$ will not be identically zero; a statistical test is used to determine whether the estimated values are close to zero. The test is based on estimating the theoretical variance of the sample estimates of $c_{3y}(m,0)$.

Sample estimates, $\hat{c}_{3y}(m, 0)$, and their variances are estimated for m ranging from `qmin` to `qmax`.

For an MA(q) process, the asymptotic variance of the sample estimate of $c_{3y}(q + 1,0)$ is given by,

$$\sigma^2(q+1) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=-q}^{2q+1}\left(1-\frac{|j|}{N}\right)$$

$$[y^2(i)y(i+q+1) - \hat{c}_{3y}(q+1,0)]$$

$$[y^2(i+j)y(i+j+q+1) - \hat{c}_{3y}(q+1,0)]$$

where N is the length of the time series.

The sample estimates are asymptotically normal and unbiased: hence, the threshold t_c in

$$\Pr\{|\hat{c}_{3y}(m+1,0)| \le t_c(m+1)\} = 1 - \text{pfa}$$

is given by

$$t_c(m+1) = inverf(1-\text{pfa}) * \sqrt{2 * \sigma^2(m+1)}$$

where `erfinv` is the MATLAB inverse error function. Let m_o denote the largest value of m in the range `qmin` to `qmax` for which $|c_{3y}(m+1,0)| > t_c(m+1)$ (so that the hypothesis of MA(m_o) model fails). Then, the MA order is declared to be $m_o + 1$. In the example, m_o is 2, and the MA order was declared to be 3.

This is a statistical test, and `pfa` specifies the fraction of the time that the test results will be wrong. In other words, in a Monte Carlo simulation of 1000 trials, you should expect the test results to be wrong $1000 * \text{pfa}$ times. Additionally note that the test condition is necessary, but not sufficient.

See Also

`arorder`

Reference

[1] Giannakis, G.B. and J.M. Mendel, “Cumulant-based order determination of non-Gaussian ARMA models,” *IEEE Trans. ASSP*, pp. 1411-21, Aug. 1990.

Purpose

Nonparametric magnitude and phase retrieval using the Matsuoka-Ulrych algorithm

Syntax

```
hest = matul(bspec)
```

Description

The phase and log magnitude of the transfer function are estimated via the Matsuoka-Ulrych algorithm, and then converted to the time-domain impulse response.

`bspec` is the bispectrum array (such as that computed by `bispeci` or `bispecd`).

`hest` is the estimated impulse response.

Algorithm

The phase unwrapping algorithm in [2] is used to resolve the phase ambiguity in the basic algorithm reported in [1].

See Also

`bispecd`, `bispeci`, `biceps`

References

[1] Matsuoka, T. and T.J. Ulrych, "Phase estimation using the bispectrum," *Proc. IEEE*, Vol. 72, pp. 1403-11, 1984.

[2] Rangoussi, M. and G.B. Giannakis, "FIR modeling using log-bispectra: Weighted least-squares algorithms and performance analysis," *IEEE Trans. Cir Sys*, Vol. 38, pp. 281-96, 1991.

nlgen

Purpose

Computes the output of a second-order Volterra system

Syntax

```
y = nlgen(x,h,q)
```

Description

Computes the output of a second-order Volterra system.

`x` is the input to the nonlinear system; it may be a vector or a matrix; if it is a matrix, columns are assumed to correspond to different realizations, and each column is processed separately.

`h` is the impulse response of the linear part; it should be a vector.

`q` is the impulse response of the quadratic part; it should be a matrix. Note that functions `nltick` and `nlpow` assume that q is symmetric.

`y` is the output of the nonlinear system, and will have the same dimensions as `x`; it is computed via,

$$y(n) = \sum_{k=0}^{L-1} h(k)x(n-k) + \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} q(k,l)x(n-k)x(n-l),$$

where L is the length of `h`, and `q` is M-by-N .

The data files `nl1.mat` and `nl2.mat` were generated via this function.

See Also

`nltick`, `nlpow`

Purpose

Nonparametric second-order Volterra System Identification method for arbitrary input signals

Syntax

```
[h, q] = nlpow(x,y,nfft)
```

Description

Implements the nonparametric method of Powers *et al* [1] for the identification of a second-order Volterra system, given both the input and output signals.

`x` is the input to the nonlinear system.

`y` is the output process; it must have the same dimensions as `x`. If `y` is a matrix, columns are assumed to correspond to independent realizations.

`nfft` is the FFT length to be used; the default value is the power of 2 greater than the length of the time series (row dimension for matrices).

`h` is the estimated impulse response of the linear part.

`q` is the estimated impulse response of the quadratic part.

Algorithm

The frequency-domain input-output relationship of the second-order Volterra system under consideration is,

$$Y(f) = H(f)X(f) + \sum_{f_1 + f_2 = f} Q(f_1, f_2)X(f_1)X(f_2)$$

It is assumed that $Q(f_1,f_2) = Q(f_2,f_1) = Q^*(-f_1,-f_2)$.

Transfer functions $H(f)$ and $Q(f_1,f_2)$ are obtained as the least squares solution to the set of equations, $Y(m) = A^T\mathbf{b}(m)$, where,

$$\mathbf{b}(m) = \left[H(m), Q\left(\frac{m+1}{2}, \frac{m-1}{2}\right), Q\left(\frac{m+3}{2}, \frac{m-3}{2}\right), \ldots, Q\left(\frac{M}{4}, m - \frac{M}{4}\right)\right]^T$$

and

$$A = \left[X(m), X\left(\frac{m+1}{2}\right)X\left(\frac{m-1}{2}\right), X\left(\frac{m+3}{2}\right)X\left(\frac{m-3}{2}\right), \ldots X\left(\frac{M}{4}\right)X\left(m - \frac{M}{4}\right)\right]^T.$$

Here, M is the FFT length `nfft`, and $m = 0, \ldots, M/2$.

See Also

`nltick`

Reference

[1] Powers, E.J., Ch.P. Ritz, C.K. An, S.B. Kim, R.W. Miksad and S.W. Nam, "Applications of digital polyspectral analysis to nonlinear systems modeling and nonlinear wave phenomena," *Proc. Workshop on Higher-Order Spectral Analysis*, pp. 73-77, Vail, Colorado, June 1989.

Purpose

Nonparametric second-order Volterra System Identification Method for Gaussian input signals

Syntax

```
[h, q] = nltick(x,y,nfft,wind,samp_seg,overlap)
```

Description

Implements Tick's nonparametric method for the identification of a second-order Volterra system, given both inputs and outputs. The inputs are assumed to be Gaussian.

`x` is the input to the nonlinear system.

`y` is the output process; it must have the same dimensions as `x`. If `y` is a matrix, columns are assumed to correspond to independent realizations.

`nfft` is the FFT length for computing power spectra and cross-bispectra; the default value is the power of 2 greater than the length of the time series (row dimension for matrices).

`wind` specifies the frequency-domain smoothing window. If `wind` is a scalar, the Rao-Gabr window of length `wind` will be used. This window is defined by [2],

$$W(m,n) = \frac{\sqrt{3}}{\pi^3}\left[1 - \frac{m^2 + n^2 + mn}{N^2}\right], \quad (m,n) \in G$$

where N is half the FFT length, `nfft`, and G is the set of points, (m,n), satisfying,

$$m^2 + n^2 + mn <= \frac{\text{wind}^2}{(\text{nfft}/2)^2}$$

- A unity value for `wind` results in no windowing.
- If `wind` <= 0, the default value of 5 will be used.
- If `wind` is a vector, it is assumed to specify a 1-D window from which a 2-D window is computed, $W(m,n) = w(m)w(n)w(m+n)$ [1]-[2].
- If `wind` is a 2-D matrix, it is assumed to specify the 2-D smoother directly. The bispectrum estimate averaged across records is smoothed by convolving with the 2-D window function. The window function should be real and nonnegative.

`samp_seg` specifies the number of samples per segment. The default value is set such that eight (possibly overlapped) records are obtained.

`overlap` specifies the percentage overlap between segments. The default value is 50. The allowed range is [0,99].

`h` is the estimated impulse response of the linear part.

`q` is the estimated impulse response of the quadratic part.

Algorithm

The output of a second-order Volterra system is given by,

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) + \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} q(k,l)x(n-k)x(n-l)$$

The linear part, $h(k)$, is estimated in the frequency domain from,

$$S_{yx}(\omega) = H(\omega)S_{xx}(\omega)$$

where $S_{xx}(\omega)$ is the power spectrum of process $x(n)$, and $S_{xy}(\omega)$ is the cross-spectrum of the processes $x(n)$ and $y(n)$. This relationship assumes that $x(n)$ is symmetrically distributed.

The quadratic part, $q(k,l)$, is estimated in the frequency domain from,

$$S_{yxx}(\omega_1,\omega_2) = 2Q(\omega_1,\omega_2)S_{xx}(\omega_1)S_{xx}(\omega_2) + S_{xx}(\omega_2)\delta(\omega_1 + \omega_2)E\{y(n)\}$$

which is obtained under the assumption that $q(k,l) = q(l,k)$, and that $x(n)$ is Gaussian.

The cross-bispectrum, $S_{yxx}(\omega_1,\omega_2)$, is estimated via the direct (FFT) method, using function `bispecdx`.

See Also

`nlpow`, `bispecdx`

Reference

[1] Tick, L.J., “The estimation of transfer functions of quadraticsystems,” *Technometrics*, 3, 563-67, Nov 1961.

Purpose

Picks peaks subject to a separation criterion

Syntax

```
[loc, val] = pickpeak(spec,npicks,rdiff)
```

Description

`pickpeak` picks the largest `npicks` peaks in the data vector `spec` such that the peaks are separated by at least `rdiff` samples. The default values are `npicks` = 2 and `rdiff` = 5. If `spec` is a matrix, each column is treated independently.

`loc` is the matrix of locations (indices) of the picked peaks; the *k*th column corresponds to the *k*th column of `spec`.

`val` is the matrix of the amplitudes of the picked peaks; the *k*th column corresponds to the *k*th column of `spec`.

A 0 in location (i,j) of array `loc` (or a NaN in array `val`) indicates that the *j*th data vector has less than *i* peaks with a separation of `rdiff` or more.

qpcgen

Purpose

Generates synthetics for the quadratically phase-coupled harmonics in colored Gaussian noise problem

Syntax

```
zmat = qpcgen
zmat = qpcgen(default)
```

Description

qpcgen generates independent realizations of the signal

$$y(n) = \sum_{k=1}^{p} \sum_{i=1}^{3} \alpha_{ki} \cos(2\pi\lambda_{ki} n + \phi_{ki}) + \sum_{k=1}^{\bar{p}} \bar{\alpha}_k \cos(2\pi\bar{\lambda}_k n + \bar{\phi}_k) + g(n)$$

where $\lambda_{k3} = \lambda_{k2} + \lambda_{k1}$, and $\phi_{k3} = \phi_{k2} + \phi_{k1}$; ϕ_{k1}, ϕ_{k2}, $\overline{\phi_k}$ are mutually independent, and uniformly distributed over $[0,2\pi]$.

If qpcgen is invoked without any input arguments, then you are prompted for the length of the realizations, the number of realizations, the number of phase-coupled triplets (p), their frequencies (λ_{k1} and λ_{k2}) and amplitudes (α_{ki}), as well as the corresponding parameters in the uncoupled case (the terms with an overbar), and the variance of the colored Gaussian noise, $g(n)$. The colored noise is generated by passing a white Gaussian noise sequence through an user-specified ARMA filter (you are prompted for these ARMA parameters). The phases ϕ_{k1}, ϕ_{k2}, and $\bar{\phi}_{ki}$, i = 1,2,3 are chosen randomly for each realization. Note that noise-free realizations can be obtained by specifying a value of zero for the noise variance.

If the function is invoked as `qpcgen(default)`, where the variable `default` may take on *any* value(s), then, the default settings are used (see Examples below).

Each column of `zmat` corresponds to a different realization.

The matrix `zmat` in the file qpc.mat can be regenerated via `zmat = qpcgen(1);`.

Purpose

Detection of quadratic phase coupling using the TOR method

Syntax

```
[arvec,bspec] = qpctor(y)
[arvec,bspec] = qpctor(y,maxlag,arorder,nfft,samp_seg, . . .
                       overlap,flag)
```

Description

`qpctor` detects quadratically phase coupled harmonics using the TOR method.

`y` is the data matrix; each column of `y` is assumed to correspond to a different realization.

`maxlag` specifies the maximum number of third-order cumulant lags, $C_{3x}(m,m)$, to be used.

`ar_order` specifies the AR order to use.If this parameter is not specified, or is not positive, a plot of the singular values of the cumulant matrix is displayed, and you are prompted to choose an order.

`nfft` specifies the FFT length; its default value is 64.

`samp_seg` specifies the number of samples per segment; the default value is the length of the time series, or the row dimension if `y` is a matrix.

`overlap` specifies the percentage overlap between segments; maximum allowed value is 99; default value is 0; the parameter is ignored if `y` is a matrix.

If `flag` is biased, then biased sample estimates of cumulants are computed (`default`); if the first letter is not 'b', unbiased estimates are computed.

`arvec` is the vector of estimated AR parameters.

`bspec` is the estimated parametric bispectrum.It is an `nfft/2`-by-`nfft` array whose upper-left handcornercorresponds to the origin in the bispectral plane; the frequenciesincrease downwards and to the right.

Algorithm

Details of the algorithm are given in the “Tutorial”.

Reference

[1] Raghuveer, M.R., and C.L. Nikias, “Bispectrum estimation: a parametric approach,” *IEEE Trans. ASSP*, Vol. 33, pp. 1213-30, Oct. 1985.

rivdl

Purpose

Adaptive AR parameter estimation using the double lattice form of the recursive instrumental variable algorithm

Syntax

```
[arvec, fref, bref, fpe] = rivdl(y)
[arvec, fref, bref, fpe] = rivdl(y,morder,arorder, . . .
                           lambda,delta,thres,nsmuth)
```

Description

The recursive instrumental variable double lattice algorithm is applied to estimate the AR parameters of a possibly nonstationary time series.The model is assumed to be causal.

`y` is the time series (must be a vector).

`morder` specifies the cumulant-order to be used; it should be 2, 3, or 4. The default value is 4.

`arorder` is the AR order (the number of stages in the lattice). The default value is 2.

`lambda` is the forgetting parameter; 0 < `lambda` <= 1. The default value is 0.998.

`delta` is the initialization value for $F(0)$ and $B(0)$. Its default value is 0.01.

`thres` is the threshold check for division by zero.If $|x|$ < `thres`, $1/x$ is set to zero. The default value is 0.0001 [1].

`nsmuth` is the window length for estimating the “steady-state” AR parameters. Its default value is min(length(y)/4, 50).

`arvec` is the steady-state AR parameter vector of length `arorder + 1`. The last `nsmuth` samples of the reflection coefficients, `fref` and `bref`, are averaged, and then converted to the AR parameters.

`fref` is an `nsamp` × `arorder` array containing the forward reflection coefficients of the upper lattice,where `nsamp` is the length of the time series `y`.

`bref` is an `nsamp` × `arorder` array containing the backward reflection coefficients of the upper lattice.

`fpe` is the final prediction error.

Algorithm

Details of the algorithm are given in the “Tutorial”.

See Also

`ivcal`, `rivtr`

References

[1] Swami, A., and J.M. Mendel, "Adaptive Cumulant-Based Estimation of ARMA Parameters," *Proc. Amer. Control Conf., ACC-88*, Atlanta, GA, pp. 2114-19, June 1988.

[2] Porat, B., B. Friedlander, and M. Morf, "Square-root covariance ladder algorithms," in *Proc. ICASSP-81*, Atlanta, GA, pp. 877-880, March 1981.

rivtr

Purpose

Adaptive AR parameter estimation using the transversal form of the recursive instrumental variable algorithm

Syntax

```
[arvec,fpe,wt] = rivtr(y)
[arvec,fpe,wt] = rivtr(y,morder,arorder,lambda,delta,nsmuth)
```

Description

The transversal form of the recursive instrumental variable algorithm isapplied to estimate the AR parameters of a possibly nonstationary time series.The model is assumed to be causal.

`y` is the time series (must be a vector).

`morder` specifies the cumulant-order to be used; it should be 2, 3, or 4. The default value is 4.

`arorder` is the AR order; the default value is 2.

`lambda` is the forgetting parameter; 0 < `lambda` <= 1. The default value is 0.998.

`delta` is the initialization value for $F(0)$ and $B(0)$. Its default value is ± 1, where the sign is the sign of the cross-correlation between the time series y and the corresponding instrumental variable, `z`, corresponding to order `morder` and unity value for `lambda` (`z` is computed via `z = ivcal(y,morder,1)`).

`nsmuth` is the window length for estimating the "steady-state" AR parameters. Its default value is min(length(y)/4, 50).

`arvec` is the AR parameter vector corresponding to the steady-state (final) weight vector, and has length `arorder + 1`. The last `nsmuth` samples of the time-varying weights, `wt`, are averaged, and then converted to the AR parameters.

`fpe` is the final prediction error.

`wt` is a `nsamp` × `arorder` array of the estimated weights of the adaptive filter as a function of time; here `nsamp` is the length of the time series, `y`.

Algorithm

Details of the algorithm are given in the "Tutorial".

See Also

`ivcal`, `rivdl`

References

[1] Swami, A., and J.M. Mendel, "Lattice Algorithms for Recursive Instrumental Variable Methods," *USC-SIPI Report-117*, University of Southern California, Los Angeles, Dec. 1987.

[2] Swami, A., *System Identification Using Cumulants*, Ph.D. Dissertation, University of Southern California, pp. 107-8, 1988.

rpiid

Purpose

Generates i.i.d. random sequence

Syntax

```
u = rpiid(nsamp)
u = rpiid(nsamp,in_type,pspike)
```

Description

`rpiid` generates a sequence of `nsamp` i.i.d. random variables, of the type described by `in_type`.

`in_type`	density function
`"exp"`	Single-sided exponential
`"lap"`	Double-sided exponential (Laplacian)
`"nor"`	Normal (Gaussian)
`"bga"`	Bernoulli-Gaussian
`"uni"`	Uniform

The default distribution is "`nor`."

`p_spike` specifies the event probability for the Bernoulli-Gaussian distribution; the default value is 0.1.

The output sequence, `u`, is normalized to zero-mean, by subtracting the theoretical (not the sample) mean. Note that the mean of *any* vector `u` can be set to a and its standard deviation to b via,

$$u = a + b * (u - \text{mean}(u))/\text{std}(u).$$

Purpose

Time-delay estimation (TDE) using the parametric third-order cross-cumulant method

Syntax

```
[delay,avec] = tde(x,y,max_delay)
[delay,avec] = tde(x,y,max_delay,samp_seg,svdflag)
```

Description

The time delay between two signals, possibly corrupted with spatially correlated colored Gaussian noise, is estimated using the parametric third-order cross-cumulant method.

`x` and `y` are the signals at the two sensors; they must have identical dimensions; if they are matrices, it is assumed that the columns correspond to different realizations.

`max_delay` is the absolute value of the maximum expected delay.

`samp_seg` specifies the number of samples per segment for estimating of cumulants; it defaults to the length of the time series (row dimension if `x` is a matrix).

`svdflag` is an optional parameter with a default value of zero; if its value is not zero, the SVD of the cumulant matrix (see the Algorithm section for details) is computed, and you are prompted to choose an order for the SVD low-rank approximation. The singular values of the cumulant matrix associated with the TDE problem rarely show a clean break (in contrast with those associated with the harmonic retrieval problem, solved by `harmest`, or the DOA problem, solved by `doa`); it is recommended that the order chosen for the SVD low-rank approximation be greater than the value of `max_delay`.

`delay` is the estimated delay of signal `y` with respect to signal `x`.

`avec` is the estimated parameter vector; it is a vector of length 2-by-`max_delay` + 1, and corresponds to time samples, –`max_delay`, . . . , `max_delay`. The estimated delay is the time sample at which `avec` attains its maximum absolute value.

Algorithm

Let $x(n) = s(n) + w_1(n)$, and $y(n) = As(n - D) + w_2(n)$, denote the two observed signals, where A is the amplitude gain, D is the signal delay between the two sensors, and $w_1(n)$ and $w_2(n)$ are assumed to be jointly Gaussian. Let P be the maximum expected delay. Then,

$$y(n) = \sum_{i=-P}^{P} a(i)x(n-i) + w(n)$$

where $a(n) = 0$, $n \neq D$, and $a(D) = A$. Let
$C_{yxx}(\tau,\rho) := E\{y(n)x(n+\tau)x(n+\rho)\}$,
and $C_{yxx}(\tau,\rho) := \mathrm{E}\{x(n)x(n+\tau)x(n+\rho)\}$. We obtain the third-order recursion,

$$C_{yxx}(\tau, \rho) = \sum_{i=-P}^{P} a(i)C_{xxx}(\tau + i, \rho + i).$$

Using this equation for various values of ρ and τ, we get a system of linear equations in the a(i)'s, $\mathbf{C}_{xxx}\mathbf{a} = \mathbf{c}_{yxx}$. The estimated delay is the index n, which maximizes $|a(n)|$. A low rank approximation of the cumulant matrix $\mathbf{C}_{xxx}$ may be used.

See Also

`tdeb`, `tder`

Reference

[1] Nikias, C.L. and R. Pan, "Time delay estimation in unknown Gaussian spatially correlated noise," *IEEE Trans. ASSP*, Vol. 36, pp. 1706-14, Nov. 1988.

Purpose

Time-delay estimation (TDE) using the conventional bispectrum method

Syntax

```
[delay,h] = tdeb(x,y,max_delay,nfft,wind,samp_seg,overlap)
```

Description

The time delay between two signals, possibly corrupted by colored Gaussian noise, is estimated using the conventional bispectrum method. This involves computing the third-order hologram (described below) that exhibits a peak at the true delay.

`x` and `y` should both be vectors or matrices with identical dimensions. If they are matrices, each column is assumed to be a different realization. These are the signals at the two sensors. The signals are assumed to have nonzero third-order cumulants; for example, they cannot be symmetrically distributed.

`max_delay` is the absolute value of the maximum expected delay.

`nfft` specifies the FFT size to be used; the nominal default value is 128; the actual FFT size used will be `max(samp_seg, nfft)`.

`wind` specifies the frequency-domain smoothing window. If `wind` is a scalar, the Rao-Gabr window [2]

$$W(m,n) = \frac{\sqrt{3}}{\pi^3}\left[1 - \frac{m^2 + n^2 + mn}{N^2}\right], \quad (m,n) \in G$$

of length `wind` will be used; here N is half the FFT length, `nfft`, and G is the set of points, (m,n), satisfying,

$$m^2 + n^2 + mn <= \frac{\text{wind}^2}{(\text{nfft}/2)^2}.$$

- A unity value for `wind` results in no windowing.
- If `wind` <= 0, the default value of 5 will be used.
- If `wind` is a vector, it is assumed to specify a 1-D window from which a 2-D window is computed, $W(m,n) = w(m)w(n)w(m+n)$ [1]-[2].
- If `wind` is a 2-D matrix, it is assumed to specify the 2-D smoother directly. The bispectrum estimate averaged across records is smoothed by convolving with the 2-D window function. The window function should be real and nonnegative.

samp_seg specifies the number of samples per segment. The default value is the power of 2 equal to or greater than 4*max_delay+1.

overlap specifies the percentage overlap between segments. The default value is 50. The allowed range is [0,99].

If y is a matrix, the columns are assumed to correspond to independent realizations; in this case overlap is set to zero, and samp_seg is set to the row dimension.

delay is the estimated delay of signal y with respect to signal x.

h is the estimated third-order hologram (defined below); it is a vector of length nfft, corresponding to indices of –nfft/2 to nfft/2–1, where nfft is the FFT length.

Algorithm

Define auto- and cross-bispectra via,

$$B_{xxx}(\omega_1,\omega_2) = E\{X(\omega_1)X(\omega_2)X^*(\omega_1 + \omega_2)\},$$

$$B_{xyx}(\omega_1,\omega_2) = E\{X(\omega_1)Y(\omega_2)X^*(\omega_1 + \omega_2)\}.$$

The third-order hologram, $h(\tau)$, is then defined by

$$h(\tau) = \int d\omega_1 \int d\omega_2 \exp(j\omega_2\tau)\frac{B_{xyx}(\omega_1, \omega_2)}{B_{xxx}(\omega_1, \omega_2)}.$$

The absolute value of the hologram should display a strong peak at the location of the true delay. Since third-order statistics are used, the method is insensitive (in theory) to both spatially and temporally colored Gaussian noise.

Sample estimates of the auto- and cross-bispectra are obtained via routine bispecdx: the data is segmented; each segment is Fourier transformed, and the triple frequency product is computed; these products are then averaged across the suite of segments to give the final cross- and auto-bispectral estimates. In order to obtain good estimates, it is critical that the data length be much larger than the specified maximum delay. The estimated delay will range from –nfft/2 to nfft/2–1.

See Also

tde, tder

Reference

[1] Nikias, C.L. and R. Pan, “Time delay estimation in unknown Gaussian spatially correlated noise,” *IEEE Trans. ASSP*, Vol. 36, pp. 1706-14, Nov. 1988.

Purpose

Generates synthetic sequences for the time-delay estimation (TDE) problem

Syntax

```
[s1,s2] = tdegen
[s1,s2] = tdegen(default)
```

Description

`tdegen` generates the data sequences:

$$s_1(n) = x(n) + g_1(n); \quad s_2(n) = Ax(n - D) + g_2(n);$$

where $x(n)$ is a zero-mean i.i.d. random sequence, with single-sided exponential density function; $g_1(n)$ and $g_2(n)$ are zero-mean colored noise sequences; A is the amplitude gain from sensor 1 to sensor 2; and D is the delay of the signal at the second sensor with respect to that at the first sensor.

If the function is invoked without any input arguments, then you are prompted for all parameters; otherwise, default settings are used.

If the function is invoked as `tdegen(default)`, where the variable `default` may take on *any* value(s), then, the default settings are used.

The colored noise sequence, $g_1(n)$, is obtained by passing an i.i.d. sequence (Laplace, uniform, or normally distributed), through an ARMA filter. The colored noise sequence, $g_2(n)$, is obtained by passing the noise, $g_1(n)$, through another ARMA filter. Note that the two noise sequences are correlated with each other.

Noise-free signals can be generated by specifying a noise variance of zero when prompted.

Vectors `s1` and `s2` contain the simulated signals at the two sensors.

The vectors `s1` and `s2` in the file `tde1.mat` can be regenerated via, `[s1,s2] = tdegen(1);`.

tder

Purpose

Time-delay estimation (TDE) using the ML-windowed cross-correlation method

Syntax

```
[delay,rxy] = tder(x,y,max_delay,samp_seg,overlap,nfft)
```

Description

The time delay between two signals, possibly corrupted by colored Gaussian noise, is estimated using the Maximum Likelihood (ML) windowed cross-correlation method.

`x` and `y` should both be vectors or matrices with identical dimensions. If they are matrices, each column is assumed to be a different realization. These are the signals at the two sensors.

`max_delay` is the absolute value of the maximum expected delay.

`samp_seg` specifies the number of samples per segment; the default value is the power of 2 just greater than or equal to `4*max_delay + 1`.

`overlap` specifies the percentage overlap between segments. The default value is 50. The allowed range is [0,99].

If `x,y` are matrices, `samp_seg` is set to the row dimension of `x` and `overlap` is set to zero.

`nfft` specifies the FFT length to use; the default value is the power of 2 just greater than `samp_seg`.

`delay` is the estimated delay of signal *y* with respect to signal *x*.

`rxy` is the estimate of the windowed autocorrelation (described below); it is a vector of length `nfft`, corresponding to indices of `–nfft/2` to `nfft/2–1`.

Algorithm

Let $S_{xy}(\omega)$ denote the cross-spectrum between the two signals, *x* and *y*; and let $S_{xx}(\omega)$ and $S_{yy}(\omega)$ denote the auto-spectra of *x* and *y*. The squared coherence function is defined by

$$C_{xy}(\omega) = \frac{\left|S_{xy}(\omega)\right|^2}{S_{xx}(\omega)S_{yy}(\omega)}.$$

The optimal-ML window is then

$$W(\omega) = \frac{1}{|S_{xy}(\omega)|}\frac{C_{xy}(\omega)}{1 - C_{xy}(\omega)},$$

and the windowed cross-correlation, $R_{xy}(m)$, is the inverse Fourier transform of $W(\omega)S_{xy}(\omega)$.

Estimates of S_{xx}, S_{xy} and C_{xy} are obtained via the Signal Processing Toolbox routine `spectrum` [R1]; the data is segmented, and spectrum estimates from individual segments are averaged; the segment length is taken to be the smallest power of 2 larger than twice the maximum delay. Since good estimates demand a large number of segments, it is critical that the lengths of the time series, *x* and *y*, be much larger than `max_delay`. The estimated delay will range from `–nfft/2` to `nfft/2–1`.

The raw delay, *d*, is estimated as the location of the peak of $|R_{xy}(m)|$. A three-point interpolation is used to improve the delay estimate:

$$d_i = d - \frac{1}{2}\frac{R_{xy}(d+1) - R_{xy}(d-1)}{R_{xy}(d+1) - 2*R_{xy}(d) + R_{xy}(d-1)}.$$

See Also

`tde`, `tdeb`

References

[1] Krauss, T., J.N. Little, and L. Shure, *MATLAB Signal Processing Toolbox User's Guide*, The MathWorks Inc., 1994.

[2] Nikias, C.L. and R. Pan, "Time delay estimation in unknown Gaussian spatially correlated noise," *IEEE Trans. ASSP*, Vol. 36, pp. 1706-14, Nov. 1988.

tls

Purpose

Total Least Squares solution to a system of linear equations

Syntax

```
[x, flag] = tls(A,b)
```

Description

`tls` obtains the Total Least Squares (TLS) solution to the linear system of equations, `Ax = b`. If `A` is m -by- n, and `b` is m-by- k, it is required that $m \geq n + k$ (that is, in the usual case, where `b` is a vector, we have $k = 1$, so that the system of equations should be overdetermined). If the problem does not have a unique solution, `flag` will be set to unity. TLS is useful when both `A` and `b` have "errors." For example, sample estimates of cumulants have errors in them due to finite record lengths.

Algorithm

The least squares solution, x_{ls}, to the set of linear equations $Ax \approx b$ minimizes the norm of the residual vector $r = Ax - b$, subject to the condition that $b + r$ is in the range space of `A`; the implicit assumption is that `A` is "clean," but `b` is noisy.

The TLSquares solution [1] assumes that both `A` and `b` may be noisy; the TLS solution, if one exists, minimizes the Frobenius norm of the matrix `[E r]` subject to the condition that $b + r$ is in the range space of $A + E$. Here, `E` and `A` have the same dimensions.

Let `A` be $m \times n$, b be $m \times k$, and $m \geq n + k$. Consider the SVD of the composite matrix [Ab] = $U\Sigma V'$ where Σ is the diagonal matrix of singular values, $\sigma(t)$, t = 1, . . ., $n + k$. In [1], it is shown that if $s(n) > s(n + 1)$, then the TLS solution exists, is unique, and is given by

$$x = -V_{12}V_{22}^{-1},$$

where the n -by-n matrix and the k -by-k matrix V22 are defined by

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}.$$

Weighted solutions and other details are discussed in [1].

Reference

[1] Golub, G.H. and C.F. Van Loan, *Matrix Computations*, pp. 420-5, Baltimore: The Johns Hopkins University Press, 1983.

Purpose

Estimates the AR parameters and the reflection coefficients given a slice of a cumulant sequence

Syntax

```
[amat,cmat,pf,gamf,gamb] = trench(c,r)
```

Description

Given a nonsymmetric Toeplitz matrix, A, described by the first column vector, c, and the first row vector, r, obtains the solutions to the set of equations

$$A_{k,k}a_k = [P_k,0]',$$

$k = 1, \ldots, M$, where $M + 1$ is the length of vectors **c** and r, and $A_{k,k}$ is the leading $k \times k$ submatrix of A. If r is not specified, it is set to c′, the conjugate transpose of c.

The typical situation is the estimation of AR parameters based on correlation or cumulant sequenes.

amat is the matrix of estimated forward AR prediction vectors, for orders 1 through M; the kth column corresponds to the AR(k) model.

cmat is the matrix of estimated backward AR prediction vectors, for orders 1 through M; the kth column corresponds to the AR(k) model.

pf is the final prediction error variance, for orders 1 through M;

gamf is the vector of forward reflection coefficients.

gamb is the vector of backward reflection coefficients.

Algorithm

The standard Trench algorithm [2] is implemented; the Levinson-Durbin algorithm [1] is obtained when r = c′.

References

[1] Haykin, S., *Adaptive Filter Theory*, New Jersey: Prentice-Hall, pp. 198-205, 2nd ed., 1991.

[2] Zohar, S., "Toeplitz Matrix Inversion: the Algorithm of W.F. Trench," *J. Assoc. Comp. Mach.*, Vol. 16, pp. 592-601, 1968.

trispect

Purpose

Computes a 2-D slice of the theoretical trispectrum of an ARMA process

Syntax

```
[tspec, waxis] = trispect(ma,ar,nfft,f3)
```

Description

A 2-D slice of the theoretical trispectrum corresponding to an ARMA process is computed.

`ma` is the MA parameter vector, and must be specified.

`ar` is the AR parameter vector; its default value is [1].

`nfft` is the FFT length to use; its default value is 512.

`f3` specifies the fixed value of the third-frequency, f_3, of the trispectrum, S3(f_1,f_2,f_3). The default value is 0; the nominal range is [-0.5,0.5]; values outside this range are folded back into it.

`tspec` is the 2-D slice of the trispectrum corresponding to the ARMA model, with fixed f_3 = `f3`. It is an `nfft`-by-`nfft` array, with origin at the center, and axes pointing down and to the right.

`waxis` is the set of frequencies associated with the `tspec`; thus, the *i*th row (or column) of `tspec` corresponds to the frequency `waxis(i)`. Frequencies are normalized; that is, the sampling frequency is assumed to be unity.

Algorithm

Let $H(f) = B(f)/A(f)$ denote the transfer function of the ARMA filter; then, the trispectrum is given by,

$$S_3(f_1,f_2,f_3) = H(f_1)H(f_2)H(f_3)H^*(f_1 + f_2 + f_3).$$

In this routine, frequency f_3 is fixed at the specified value of `f3`.

See Also

`bispect`, `cumtrue`

References

[1] Subba Rao, T. and M. Gabr, *An Introduction to Bispectral Analysis and Bilinear Time-Series Models*, pp. 42-43, New York: Springer-Verlag, 1984.

[2] Nikias, C.L., and M.R. Raghuveer, “Bispectrum estimation: a digital signal processing framework,” *Proc. IEEE*, Vol. 75, pp. 869-91, July 1987.

Purpose

Computes the Wigner spectrum

Syntax

```
[wx, waxis] = wig2(x,nfft,flag)
```

Description

Estimates the Wigner spectrum (WS) of a signal using the time-domain approach.

`x` is the signal and *must* be a vector.

`nfft` specifies the FFT length, and hence, the frequency resolution; this parameter must be greater than twice the length of the signal, `x`, in order to avoid aliasing. The default value is the power of 2 equal to or just greater than twice the signal length.

`flag` – if `flag` is nonzero and `x` is real-valued, the analytic form of `x` is used instead of `x`; the default value is 1. If $x(t)$ is a real-valued signal, with Hilbert transform $y(t)$, then, the analytic signal is defined as the complex-valued signal $z(t) = x(t) + jy(t)$.

`wx` is the computed WS. The rows correspond to time samples, and the columns to frequencies.

`waxis` is a vector whose entries are the frequencies associated with the columns of the WS.

Algorithm

Let the instantaneous autocorrelation be defined by

$$r(m,n) = x^*(n - m)x(n + m)$$

where n is identified with time and m with lag. The WS is then given by

$$W(f,n) = \sum_m r(m, n)\exp(-j\pi f m).$$

The original signal must be sampled at twice the Nyquist rate or faster, in order to avoid aliasing.

See Also

`wig2c`, `wig3`, `wig3c`, `wig4`, `wig4c`

References

[1] Hlaswatch, F., and G.F. Boudreaux-Bartels, "Linear and Quadratic Time-Frequency Representations," *IEEE Signal Processing Magazine*, pp. 21-67, Apr. 1992.

[2] Cohen, L., "Time-Frequency Distributions: A Review," *Proc. IEEE*, pp. 941-81, July 1989.

wig2c

Purpose

Computes the Wigner spectrum with Choi-Williams filtering

Syntax

```
[wx, waxis] = wig2c(x,nfft,sigma,flag)
```

Description

Estimates the Wigner spectrum (WS) of a signal; eliminates cross-terms in the WS of multi-component signals by applying the Choi-Williams filter.

`x` is the signal and *must* be a vector.

`nfft` specifies the FFT length, and hence, the frequency resolution; this parameter must be greater than twice the length of the signal, x, in order to avoid aliasing. The default value is the power of 2 equal to or just greater than twice the signal length.

`sigma` is the parameter in the Choi-Williams window, and controls the amount of cross-term suppression; very large values result in no suppression, whereas very small values result in loss of signal terms. The default value is 0.05.

`flag` – if `flag` is nonzero and `x` is real-valued, the analytic form of `x` is used instead of `x`; the default value is 1. If $x(t)$ is a real-valued signal, with Hilbert transform $y(t)$, then, the analytic signal is defined as the complex-valued signal $z(t) = x(t) + jy(t)$.

`wx` is the computed WS. The rows correspond to time samples, and the columns to frequencies.

`waxis` is a vector whose entries are the frequencies associated with the columns of the WS.

Algorithm

Let the instantaneous autocorrelation be defined by

$$r(m,n) = x^*(n - m)x(n + m)$$

where n is identified with time and m with lag. The ambiguity function, AF, is then given by

$$AF(m, \theta) = \sum_m r(m, n) \exp(-j2\pi n\theta).$$

The AF is multiplied by the Choi-Williams window function,

$$w(m,\theta) = \exp(-(m\theta/\texttt{sigma})^2).$$

The 2-D FT of $AF(m,\theta)w(m,\theta)$ (θ to n and m to f) yields the filtered WS, $W_c(f,n)$.

The original signal must be sampled at twice the Nyquist rate or faster.

See Also

`wig2`, `wig3`, `wig3c`, `wig4`, `wig4c`

References

[1] Hlaswatch, F., and G.F. Boudreaux-Bartels, "Linear and Quadratic Time-Frequency Representations," *IEEE Signal Processing Magazine*, pp. 21-67, Apr. 1992.

[2] Cohen, L., "Time-Frequency Distributions: A Review," *Proc. IEEE*, pp. 941-81, July 1989.

[3] Choi, H. and W.J. Williams, "Improved Time-Frequency Representation of Multicomponent Signals Using Exponential Kernels," *IEEE Trans. ASSP*, pp. 862-71, June 1989.

wig3

Purpose

Computes the $f_1 = f_2$ slice of the Wigner bispectrum

Syntax

```
[wx, waxis] = wig3(x,nfft,flag)
```

Description

Computes the diagonal slice of the Wigner bispectrum (WB).

`x` is the signal and *must* be a vector.

`nfft` specifies the FFT length, and hence, the frequency resolution; this parameter must be greater than three times the length of the signal, `x`, in order to avoid aliasing. The default value is the power of 2 equal to or just greater than three times the signal length.

`flag` – if `flag` is nonzero and `x` is real-valued, the analytic form of `x` is used instead of `x`; the default value is 1. If $x(t)$ is a real-valued signal, with Hilbert transform $y(t)$, then, the analytic signal is defined as the complex-valued signal $z(t) = x(t) + jy(t)$.

`wx` is the computed WB. The rows correspond to time samples, and the columns to frequencies.

`waxis` is a vector whose entries are the frequencies associated with the columns of the WB.

Algorithm

Define the instantaneous triple product

$$r_3(t,\tau_1,\tau_2) = x^*(t - \alpha\tau_1 - \alpha\tau_2)x(t + \beta\tau_1 - \alpha\tau_2)x(t - \alpha\tau_1 + \beta\tau_2)$$

where $\alpha = 1/3$ and $\beta = 2/3$. The WB is then given by

$$W(n, f_1, f_2) = \iint d\tau_1 d\tau_2 e^{-j2\pi(f_1\tau_1 + f_2\tau_2)} r_3(t, \tau_1, \tau_2)$$

In this routine we compute the slice $f_1 = f_2$.

Note that the original signal should be sampled at twice the Nyquist rate, in order to avoid aliasing. Also note that the frequency axes are scaled by the factor of 2/3; in this routine, we undo the scaling so that you can find the peaks at the expected frequencies.

See Also

`wig2`, `wig2c`, `wig3c`, `wig4`, `wig4c`

References

[1] Fonollosa, J.R. and C.L. Nikias, "Wigner Higher-Order Moment Spectra: Definitions, Properties, Computation and Applications to Transient Signal Detection," *IEEE Trans. SP*, Jan. 1993.

[2] Gerr, N.L., "Introducing a third-order Wigner distribution," *Proc. IEEE*, pp. 290-92, Mar. 1988.

[3] Swami, A., "Third-order Wigner distributions," *Proc. ICASSP-91*, pp. 3081-84, Toronto, Canada, May 1991.

wig3c

Purpose

Computes the diagonal slice of the Wigner bispectrum, with Choi-Williams filtering

Syntax

```
[wx, waxis] = wig3c(x,nfft,sigma,flag)
```

Description

Computes the diagonal slice of the Wigner bispectrum (WB); the Choi-Williams filter is applied in order to suppress the cross-terms in the WB of multicomponent signals.

`x` is the signal and *must* be a vector.

`nfft` specifies the FFT length, and hence, the frequency resolution; this parameter must be greater than three times the length of the signal, `x`, in order to avoid aliasing. The default value is the power of 2 equal to or just greater than thrice the signal length.

`sigma` is the parameter in the Choi-Williams window, and controls the amount of cross-term suppression; very large values result in no suppression, whereas very small values result in loss of signal terms. The default value is 0.5.

`flag` – if `flag` is nonzero and `x` is real-valued, the analytic form of `x` is used instead of `x`; the default value is 1. If $x(t)$ is a real-valued signal, with Hilbert transform $y(t)$, then, the analytic signal is defined as the complex-valued signal $z(t) = x(t) + jy(t)$.

`wx` is the computed WB. The rows correspond to time samples, and the columns to frequencies.

`waxis` is a vector whose entries are the frequencies associated with the columns of the WB.

Algorithm

Define the instantaneous triple product

$$r_3(t,\tau_1,\tau_2) = x^*(t - \alpha\tau_1 - \alpha\tau_2)x(t + \beta\tau_1 - \alpha\tau_2)x(t - \alpha\tau_1 + \beta\tau_2)$$

where $\alpha = 1/3$ and $\beta = 2/3$. Define the smoothing kernel,

$$\Phi(\theta, \tau_1, \tau_2) = \exp(-\theta^2(r_1^2 + r_2^2)/\text{sigma})$$

The filtered WB is then given by

$$W(n, f_1, f_2) = \iiint\!\!\iint d\theta\, d\tau_1\, d\tau_2\, du\; e^{-j2\pi(f_1\tau_1 + f_2\tau_2)} e^{-j2\pi t\theta} e^{j2\pi u\theta}$$
$$r_3(t, \tau_1, \tau_2)\Phi(\theta, \tau_1, \tau_2)$$

In this routine we compute the slice $f_1 = f_2$.

Note that the original signal should be sampled at twice the Nyquist rate, in order to avoid aliasing. Also note that the frequency axes are scaled by the factor of 2/3. In this routine, we undo the scaling so that you can find the peaks at the expected frequencies.

NOTE: The application of the Choi-Williams kernel to the Wigner bispectrum does not guarantee preservation of the auto-terms; consequently, this routine should be used with great caution.

See Also

`wig2`, `wig2c`, `wig4`, `wig4c`

References

[1] Choi, H. and W.J. Williams, "Improved Time-Frequency Representation of Multicomponent Signals Using Exponential Kernels," *IEEE Trans. ASSP*, pp. 862-71, June 1989.

[2] Fonollosa, J.R. and C.L. Nikias, "Wigner Higher-Order Moment Spectra: Definitions, Properties, Computation and Applications to Transient Signal Detection," *IEEE Trans. SP*, Jan. 1993.

wig4

Purpose

Computes the $f_1 = f_2 = -f_3$ slice of the Wigner trispectrum

Syntax

```
[wx, waxis] = wig4(x,nfft,flag)
```

Description

Estimates the $f_1 = f_2 = -f_3$ slice of the Wigner trispectrum (WT) of a signal using the time-domain approach.

`x` is the signal and *must* be a vector.

`nfft` specifies the FFT length, and hence, the frequency resolution; this parameter must be greater than four times the length of the signal, x, in order to avoid aliasing. The default value is the power of 2 equal to or just greater than four times the signal length.

`flag` – if `flag` is nonzero and `x` is real valued, the analytic form of `x` is used instead of `x`; the default value is 1. If $x(t)$ is a real-valued signal, with Hilbert transform $y(t)$, then, the analytic signal is defined as the complex-valued signal $z(t) = x(t) + jy(t)$.

`wx` is the computed WT. The rows correspond to time samples, and the columns to frequencies.

`waxis` is a vector whose entries are the frequencies associated with the columns of the WT.

Algorithm

For a complex signal, the Wigner trispectrum can be defined in two different ways; here, we implement the *symmetric* Wigner trispectrum [1].

Define the instantaneous fourth-order product,

$$r_4(t,\tau_1,\tau_2,\tau_3) = x^*(t-\tau)x(t-\tau+\tau_1)x(t-\tau+\tau_2)x^*(t-\tau+\tau_3)$$

where $\tau := (\tau_1 + \tau_2 + \tau_3)/4$. Notice that two of the terms are conjugated.

The symmetric Wigner trispectrum (WT) is then given by

$$W(n, f_1, f_2, f_3) = \iiint d\tau_1 d\tau_2 d\tau_3 e^{-j2\pi(f_1\tau_1 + f_2\tau_2 + f_3\tau_3)} r_4(t, \tau_1, \tau_2, \tau_3)$$

In this routine we compute the slice $f_1 = f_2 = -f_3 = f$.

Note that the original signal should be sampled at twice the Nyquist rate, in order to avoid aliasing. Also note that the frequency axes are scaled by the

factor of 1/2; in this routine, we undo the scaling so that you can find the peaks at the expected frequencies.

See Also

`wig2`, `wig2c`, `wig3`, `wig3c`, `wig4c`

References

[1] Fonollosa, J.R. and C.L. Nikias, "Analysis of finite-energy signals using higher-order moments- and spectra-based time-frequency distributions," *Signal Processing*, Vol. 36, pp. 315-28, 1994.

[2] Swami, A., "Higher-Order Wigner Distributions," in *Proc. SPIE-92, Session on Higher-Order and Time-Varying Spectral Analysis*, Vol. $1770, 290-301, San Diego, CA, July 19-24, 1992.

wig4c

Purpose

Computes the *Sliced Reduced-Interference* Wigner trispectrum

Syntax

```
[wx, waxis] = wig4c(x,nfft,sigma,flag)
```

Description

Computes the Sliced Reduced-Interference Wigner trispectrum (WT) [2]; eliminates cross-terms in the WT of multicomponent signals by applying the Choi-Williams filter.

`x` is the signal and *must* be a vector.

`nfft` specifies the FFT length, and hence, the frequency resolution; this parameter must be greater than four times the length of the signal, `x`, in order to avoid aliasing. The default value is the power of 2 equal to or just greater than four times the signal length.

`sigma` is the parameter in the Choi-Williams window, and controls the amount of cross-term suppression; very large values result in no suppression, whereas very small values result in loss of signal terms. The default value is 0.05.

`flag` – if `flag` is nonzero and `x` is real valued, the analytic form of `x` is used instead of `x`; the default value is 1. If $x(t)$ is a real-valued signal, with Hilbert transform $y(t)$, then, the analytic signal is defined as the complex-valued signal $z(t) = x(t) + jy(t)$.

`wx` is the computed WT. The rows correspond to time samples, and the columns to frequencies.

`waxis` is a vector whose entries are the frequencies associated with the columns of the WT.

Algorithm

For a complex signal, the Wigner trispectrum can be defined in two different ways; here, we implement the *symmetric* Wigner trispectrum [1].

Define the instantaneous fourth-order product,

$$r_4(t,\tau_1,\tau_2,\tau_3) = x^*(t-\tau)x(t-\tau+\tau_1)x(t-\tau+\tau_2)x^*(t-\tau+\tau_3)$$

where $\tau := (\tau_1 + \tau_2 + \tau_3)/4$. Define the smoothing kernel,

$$\Phi(\theta,\tau_1,\tau_2,\tau_3) = \exp(-\theta^2(r_1^2 + r_2^2 + r_3^2)/\text{sigma})$$

The filtered WT is then given by

$$W(n, f_1, f_2, f_3) = \iiiint d\theta d\tau_1 d\tau_2 d\tau_3 du e^{-j2\pi(f_1\tau_1 + f_2\tau_2 + f_3\tau_3)} e^{-j2\pi t\theta} e^{j2\pi u\theta} r_4(t, \tau_1, \tau_2, \tau_3)\Phi(\theta, \tau_1, \tau_2, \tau_3)$$

In this routine we compute the slice $f_1 = f_1 = -f_3$.

Note that the original signal should be sampled at twice the Nyquist rate, in order to avoid aliasing. Also note that the frequency axes are scaled by the factor of 1/2. In this routine, we undo the scaling so that you can find the peaks at the expected frequencies.

It should be noted that the Wigner trispectrum is real valued.

See Also

`wig2`, `wig2c`, `wig3`, `wig3c`, `wig4`

References

[1] Choi, H. and W.J. Williams, "Improved Time-Frequency Representation of Multicomponent Signals Using Exponential Kernels," *IEEE Trans. ASSP*, pp. 862-71, June 1989.

[2]Fonollosa, J.R. and C.L. Nikias, "Analysis of finite-energy signals using higher-order moments- and spectra-based time-frequency distributions," *Signal Processing*, Vol. 36, pp. 315-28, 1994.

Index

A

adaptive filter
  double lattice 2-72
  RIV 2-74
adaptive linear prediction 1-54
ambiguity function 1-90-1-92
AR method
  DOA 2-44
  harmonic retrieval 2-50
AR models 1-31
  order determination 1-34, 2-15
  parameter estimation 2-17
  parameter identifiability 1-31
`ar1.mat` 1-134
ARMA models 1-32
  AR order estimation 2-15
  AR parameter estimation 2-17
  residual time series 2-12
`arma1.mat` 1-135
`armaqs` 1-33, 2-8
`armarts` 2-11
`armasyn` 2-14
`arorder` 2-15
`arrcest` 2-17
autocorrelation 1-6

B

backward prediction problem 1-47
beamformer 1-64, 1-65, 1-77
bibliography 1-139
`biceps` 2-19
`bicepsf` 2-21
bicepstrum 1-38
`bicoher` 2-23
bicoherence 1-4
  auto 2-23
  cross 2-25
  estimation 1-20, 2-23
`bicoherx` 2-25
`bispecd` 2-27
`bispecdx` 2-29
`bispeci` 2-31
`bispect` 2-33
bispectrum 1-8
  cross 2-29
  direct estimate 1-19
  direct method 2-27
  estimation 1-17
  indirect method 1-18, 2-31
  theoretical 2-33
  Wigner 2-90, 2-92
Burg's maximum-entropy estimator 1-74

C

Canadian lynx data 1-114
Capon (ML) 1-74
Capon's maximum-likelihood estimator 1-73
Choi-Williams
  distributions 1-89
  filter 1-92, 1-95, 1-100
  smoothing 2-88
cross-bicoherence 1-10
cross-biperiodogram 1-17
cross-bispectra
  Volterra systems 1-80
cross-bispectrum 1-9
  direct estimate 1-18
  direct method 2-29
  estimation 1-13
  indirect estimate 1-16
cross-cumulant 1-81

cum2x 2-34
cum3x 2-36
cum4x 2-38
cumest 2-40
cumtrue 2-42
cumulants 1-4
  auto 2-40
  definitions 1-6
  fourth-order 2-40, 2-42
  sample estimates 1-12
  second-order 2-34, 2-40
  third-order 2-36, 2-41
  true 2-42

D

demos 2-54
DOA 1-62, 1-64
  AR 1-74
  beamformer 1-74
  Capon(ML) 1-74
  cumulant-based estimators 1-74
  eigenvector 1-74
  ESPRIT 1-74
  fourth-order cumulants 2-44
  minimum-norm 1-74
  MUSIC 1-74
  Pisarenko 1-74
  spatial covariance matrix 2-44
doa 2-44
doa1.mat 1-135
doagen 2-46

E

eda 1-112, 1-114, 1-120
eigenvector method
  DOA 2-44
  harmonic retrieval 2-50
eigenvector methods 1-67
ESPRIT 1-70, 1-74
  DOA 2-44
examples
  AR order determination 1-34
  AR parameter estimation 1-32
  ARMA parameter estimation 1-34
  bicepstrum-based IR estimation 1-39, 1-41
  bicoherence estimation 1-20
  computing true cumulants 1-43
  cross-bicoherence estimation 1-20
  cross-bispectrum 1-19
  cumulant estimation 1-14
  cumulation estimation 1-14
  DOA estimation 1-77
  Gaussianity-linearity tests 1-24
  harmonic retrieval 1-75
  Levinson recursion 1-50
  MA order determination 1-37
  MA parameter estimation 1-30
  Matsuoka-Ulrych algorithm 1-42
  QPC detection 1-87
  RIV double-lattice form 1-60
  RIV transversal form 1-57
  speech signal 1-122
  sunspot data 1-108
  time-delay estimation 1-103, 1-105, 1-107
  trench recursion 1-50
  Volterra system identification 1-82, 1-83
  Wigner bispectrum 1-96
    smoothed 1-97
  Wigner spectrum 1-93
    smoothed 1-94
  Wigner trispectrum 1-99
    smoothed 1-100

F

FBLS 1-53
  deterministic formulation 1-53
forward prediction problem 1-47
forward-backward least squares problem 1-54
frequency coupling 1-128
frequency estimation 1-65, 2-50

G

Gaussianity test 1-22, 2-47
`gldat.mat` 1-135
`glstat` 2-47
GM equations 1-29
guided tour 2-54

H

`harm.mat` 1-135
`harmest` 2-50
`harmgen` 2-53
harmonic retrieval 1-62, 1-64
  AR models 1-66
  ARMA models 1-66
  cumulant-based method 1-74
  minimum-norm method 1-69
  MUSIC 1-68
  Pisarenko's method 1-67
  synthetics 2-53
help 2-55
higher-order spectra 1-2
higher-order statistics 1-2
  motivations 1-10
hologram
  third order 1-106
`hosademo` 2-54
`hosahelp` 2-55
`hprony` 2-56

I

instrumental variables 2-57
`ivcal` 2-57

K

kurtosis 1-7

L

Levinson-Durbin recursion 1-48, 2-85
linear models
  frequency-domain bicepstral method 2-21
  lag-domain bicepstral method 2-19
linear prediction 1-31, 1-47
  adaptive 1-54
linear processes
  impulse response estimation 1-37
  theoretical cumulants 1-43
  theoretical polyspectra 1-43
linearity test 1-24
linearity tests 2-47

M

MA models 1-29
  order estimation 2-61
  parameters estimation 2-58
`ma1.mat` 1-135
`maest` 2-58
`maorder` 2-61
Matsuoka-Ulrych algorithm 2-63
`matul` 2-63
minimum phase 1-11
minimum-norm method
  DOA 2-44
  harmonic retrieval 2-50

mixed-phase 1-11, 1-135
ML-Capon 1-74
MUSIC 1-68
  DOA 2-44
  harmonic retrieval 2-50

N

nl1.mat 1-135
nl2.mat 1-136
nlgen 2-64
nlgen 2-64
nlpow 2-65
nltick 2-67
noise subspace 1-69
nonredundant region 1-133
normal equations
  cumulant-based 2-17
  deterministic 1-53

P

peak picking 2-69
periodogram 1-15, 1-64
phase coupling 1-84, 1-132
pickpeak 2-69
Pisarenko's method 1-67
  DOA 2-44
  harmonic retrieval 2-50
pitfalls 1-131
polycepstra methods 1-38
polycepstral methods 1-38
polycepstrum 1-40
polyspectra
  linear processes 1-4
  windows 1-16
polyspectrum
  definitions 1-6
power spectrum
  Wigner 2-87
power spectrum estimation 1-4
  conventional methods 1-15
  criterion-based estimators 1-72
  criterion-based methods 1-15
  model-based methods 1-15
  non-parametric methods 1-15
  parametric estimators
    ARMA models 1-26
power spectrum estimator
  Burg estimator 1-72
  Capon's ML estimator 1-72
  MVD estimator 1-72
Prony's method 2-56

Q

QPC 1-88
  detection 2-71
  synthetics 2-70
qpc.mat 1-137
qpcgen 2-70
qpctor 2-71
q-slice method 1-33
quadratic phase coupling 1-84
quick help 2-55

R

random sequence generator 2-76
recursive instrumental variable (RIV) algorithm 1-56
recursive least squares (RLS) algorithm 1-56
reflection coefficients 1-49, 1-60
residual time series 1-32

resolution 1-65
RIV
  double-lattice form 1-58
  transversal form 1-56
RIV algorithm 1-56
riv.mat 1-137
rivdl 2-72
rivtr 2-74
RLS algorithm 1-47
rpiid 2-76

S

self-driving AR model 1-66
signal subspace 1-68
skewness 1-7
speech signals 1-122
sunspot data 1-108
synthetic generator
  harmonics in noise 2-53
synthetics 1-65
system identification
  non-parametric 2-19, 2-21, 2-63

T

TDE
  cross-bispectral method 2-80
  cross-correlation method 1-101
  cross-cumulant method 1-101, 2-77
  ML window cross-correlation method 2-82
  synthetics 2-81
  using hologram 1-105
tde 2-77
tde1.mat 1-137
tdeb 2-79
tdegen 2-81
tder 2-82
TFD's
  Cohen class 1-89
time-delay estimation problem 1-101
time-frequency distribution 1-89
tls 2-84
total least squares 2-84
tprony.mat 1-137
transient signals 1-89
transients modeling 2-56
trench 2-85
Trench recursion 1-49
trench recursion 2-85
tricks 1-131
trispect 2-86
trispectrum 1-8
  theoretical 2-86
  Wigner 2-94, 2-96

V

variance 1-65
Volterra
  non-Gaussian inputs 1-82
Volterra models
  arbitrary inputs 2-65
  computing output 2-64
  Gaussian inputs 2-67
Volterra system 1-80

W

wig2 2-87
wig2c 2-88
wig3 2-90
wig3c 2-92
wig4 2-94

wig4c 2-96
wigdat.mat 1-137
Wigner bispectrum 1-94, 2-90
  smoothed 2-92
Wigner cross spectrum 1-90
Wigner spectrum 1-90, 2-87
  smoothed 2-88
Wigner trispectrum 1-98, 2-94
  sliced 1-98
  smoothed 2-96
Wigner-Ville distribution 1-89
window function 1-16, 1-90
Wold's decomposition 1-5